

# **COLLISION RESOLUTION ALGORITHMS FOR FINITE USER BLOCKED RANDOM ACCESS CHANNELS**

A Thesis Submitted  
in Partial Fulfilment of the Requirements  
for the Degree of

**DOCTOR OF PHILOSOPHY**

*by*  
**ABHAY KARANDIKAR**

*to the*  
**DEPARTMENT OF ELECTRICAL ENGINEERING  
INDIAN INSTITUTE OF TECHNOLOGY, KANPUR  
SEPTEMBER, 1994**

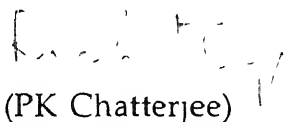


01/9/94

# Certificate

---

It is certified that the work contained in the thesis entitled "COLLISION RESOLUTION ALGORITHMS FOR FINITE USER BLOCKED RANDOM ACCESS CHANNELS", by Abhay Karandikar, has been carried out under our supervision and that this work has not been submitted elsewhere for a degree

  
(PK Chatterjee)

Professor

Department of Electrical Engineering  
Indian Institute of Technology  
Kanpur, INDIA

  
(PRK Rao)

Professor

Department of Electrical Engineering  
Indian Institute of Technology  
Kanpur, INDIA

SEPTEMBER 1994

5 3 JUL 1996  
CENTRAL LIBRARY  
I I T., KANPUR  
Acc. No. A. . . 71801

EE-1994-D-KAR-COL



A121801

# Synopsis

---

This thesis addresses itself to the problem of optimal random access algorithm for a finite user model. More specifically, we study a class of random access algorithms known as the blocked random access algorithms. Consequently, we focus only on the collision resolution aspect of a random access algorithm.

In recent times, there has been a considerable interest in the study of random multiple access (or simply random access) communication systems due to the widespread use of computer communication networks. In random access communication, there are many sources of bursty data traffic (called users) which send messages through a common broadcast communication channel. In this thesis, we consider the random access systems which can be modelled by what is called the *Common Receiver Model*. In a common receiver model, multiple users send messages to a single receiver. The common receiver model can, in fact, be considered as an instance of random access communication and after taking propagation delay into account, the random access communication model can be considered as the superposition of several such common receiver models. For our purposes, it suffices to focus on the common receiver model. The signal received at the receiver may be corrupted due to possible simultaneous transmission of messages. This interference is called collision. The collided messages require retransmission. The random access problem deals with the efficient scheduling of these messages and the scheduling strategies that the users (with bursty traffic) follow are called random access algorithms or protocols.

In general, the multiple access problem is concerned with using a common broadcast communication channel by many users. This problem has also been considered within the framework of information theory [9]. In random access

communication the emphasis is on the bursty arrival of messages. Another important difference is that in random access communication, the channel noise is ignored thus decoupling the random access communication problem from the standard communication problems studied in Shannon's sense.

The major parameters that are of interest for characterizing the efficiency of any random access algorithm are throughput, delay and the stability of the system. The messages are assumed to be broken up into segments called packets. In our study, the operation of the system is assumed to be time slotted. Thus the users can transmit their packets only at the beginning of a slot. It is also assumed that the users receive ternary valued feedback at the end of each slot. This feedback is represented by 0, 1 or 2 depending upon whether the slot contained no packet, one packet or more than packet respectively. Two models of user population have been considered in the literature. In *Infinite User Model*, there are infinite number of users and the aggregate traffic of the packets generated by all users forms a Poisson process. In *Finite User Model*, there is a fixed set of finite users and each user generates packets independent of each other. We consider the finite user model for our study.

Beginning with the development of ALOHA [1], there has been an explosion in proposing random access algorithms, each claiming better throughput than the previous one. Though the work of Capetanakis [7], Massey [44], Gallager [16], Pippenger [56], Molle [50] and Tsybakov [78] substantially improved the understanding of random access algorithms, however, as pointed out by Massey [43], the concern for proposing ever new algorithms has sometimes lead to wrong comparisons between various schemes that in fact require different channel models. Thus most of the studies in random access communication are not directed towards developing a general theory of random access algorithms. This may be, in part, due to the ever increasing pressure of network designer community to come up with practically useful protocols. One of the major contributions towards developing a theory of random access algorithms was made by Tsybakov [74], who looked at the structure of the information pattern available to the users for taking decisions and gave a formal notion of a random access algorithm for the infinite user model.

Tsybakov's work provides a mathematical framework for the study of random access algorithms for the infinite user model. This gives us the motivation to investigate whether a systematic formulation of optimal random access algorithm for the finite user model can be given. Such a formulation is not only desirable for the completeness of the theory but also can provide a framework for comparing different algorithms and establish their interrelationships. One of the important conclusions of the thesis is that an optimal algorithm for a class of random access algorithms can be (and has been) derived under a set of not quite unrealistic assumptions. We summarize some contributions of this thesis in the following.

## Main Results

This thesis has been organized in to seven chapters. In Chapter 1, we give an introduction of random access problem and a brief historical view of random access algorithms.

Following Tsybakov's work [78, 80], Chapter 2 is an attempt towards developing an unified framework for the study of random access algorithms. Accordingly, we give a systematic treatment of some random access algorithms within this framework. We consider the random access systems which can be modelled by the common receiver model and discuss this model in detail. We show how different random access algorithms can be classified using Tsybakov's definition. We then discuss various performance measures like delay, throughput, stability, backlog and average collision resolution lengths considered in the literature. The issues related to the capacity of a random access channel and an optimal algorithm are reviewed. We then discuss the operations of some random access algorithms for the infinite user model. The finite user model is considered next. Tsybakov's definition of random access algorithm for the infinite user model enables us to define a random access algorithm for the finite user model. We provide a classification of random access algorithms according to this definition and briefly review some of the existing algorithms which can be considered to be members of a particular class.

of algorithms. Finally, we consider random access systems with multipacket reception capability. This study is motivated by the fact that such channels can model code division multiple access (CDMA) systems [60] whose practical usefulness in packet radio networks has been demonstrated [82]. A class of such systems can be shown to be a generalization of the common receiver model. We therefore consider this generalized model and provide a brief sketch of the recent progress made in the study of random access algorithms for this model.

We then focus our attention on a particular class of random access algorithms for the finite user model. Our interest in this class is motivated by Tsybakov's appealing conjecture that an optimal algorithm over this class is also optimal over the entire class for the infinite user model. Another interesting aspect of the algorithms of this class is that they can be shown to consist of a channel access algorithm and a collision resolution algorithm. The channel access algorithm specifies the slot for the first time transmission of a newly generated packet, while the collision resolution algorithm is a strategy that the users follow to resolve the collision of their packets, if any. We assume the channel access to be the blocked access. The operation of the system can then be divided into successive transmission intervals called collision resolution epochs.

In Chapter 3, we consider the problem of an optimal collision resolution algorithm for the finite user model. To begin with, we assume that the number of active users at the beginning of a collision resolution epoch is known. Furthermore, the users can transmit only one packet per epoch. We identify an appropriate state of the system and control variables that constitute a Markovian Decision Process and show that the problem of optimal collision resolution algorithm is equivalent to the first passage problem of this Markovian Decision Process. The dynamic programming technique is then applied to derive an optimal collision resolution policy. The algorithm is illustrated with numerical examples. Finally, we present the steady state operation of our system using this collision resolution algorithm and suggest an approximate way to calculate the average collision resolution length.

We then remove the restriction of the number of active users at the beginning to be known and consider a more general model in Chapter 4. Thus instead of the state being known exactly, we assume that the probability distribution over the initial states is given. The conditional probability vector can be updated using the past control action and the feedback observed at each stage. It is shown that this conditional probability vector constitutes the appropriate new state and the problem of optimal collision resolution can be considered within the framework of Partially Observable Markovian Decision Process. The implementation of this optimal algorithm is, however, computationally burdensome and hence we look for reasonably good suboptimal algorithms. We suggest two suboptimal algorithms. One is based upon choosing the control corresponding to the state which has the maximum value of probability, while the other is based upon choosing the control corresponding to the expected state as if these were the actual state. The performance of these algorithms is studied by simulations with different initial probability distributions. These probability distributions are chosen to represent a range of users from lightly loaded to heavily loaded. The expected collision resolution length is then compared with a bound on the optimum performance for each initial probability distribution. The degradation in performance for the two suboptimal algorithms is not very significant. We then discuss the performance of these algorithms in the presence of feedback error by simulations.

In Chapter 5, we consider the case of users with infinite buffers. For such buffered users, we consider an access protocol which employs the optimal collision resolution algorithm for the case of single buffer. The head of the line packet of each active user participates in the collision resolution process using the collision resolution algorithm. As soon as any of these packets is transmitted successfully, the collision resolution algorithm is interrupted and those other packets of the successful user which were generated in the previous epoch are transmitted in consecutive slots. The user signals the end of the transmission of packets by an empty slot. The algorithm then again starts from the point where it was interrupted. We perform the steady state analysis of this protocol. The sequence of transmission epoch lengths is shown to be a Markov chain. The expressions for evaluating the state transition probabilities,

the steady state probabilities and the expected transmission epoch lengths are derived. This protocol is compared with TDMA scheme in terms of *throughput* and another measure of performance called *resolution factor* which we have suggested. The average packet delay analysis is then performed. It is shown that the packet delay consists of three components. The expressions for evaluating each of these components have been derived. The values of packet delays evaluated numerically for different arrival probabilities for a fixed set of users are presented. We then consider a type of polling protocol and perform a similar analysis for this protocol. This protocol is in fact a variant of TDMA for buffered user. The two protocols are compared with respect to the expected packet delay and the superiority of the protocol employing collision resolution algorithm is demonstrated.

Finally in Chapter 6, we study random access systems with multipacket reception capabilities. We consider a generalized common receiver model with transmitter based code assignment. Each transmitter is provided with a set of orthogonal codes. The number of such codes is much smaller than the total number of users in the system. Each user, before transmitting its packet, selects one of these codes as its address and then encodes its packet with this address. If two or more users choose the same code, then their packets collide. We extend the formulation of the optimal collision resolution algorithm to such systems. The collision resolution algorithm specifies the number of users to be enabled for each code. The dynamic programming equations are solved to derive an optimal algorithm. The algorithm is illustrated with numerical examples.

In Chapter 7, we summarize important conclusions of the thesis and indicate some directions for future research.

*Dedicated to my Parents—  
Aai and Baba*

# Acknowledgements

---

In the Indian social context, it has indeed been a difficult decision to leave a 'good' job and join back the academics and even more difficult to sustain the enthusiasm throughout the program. And if it has been possible for an ordinary, mediocre and impatient person like me, it is due to the constant encouragement and support, I got from my enthusiastic and competent friend Ajit Chaturvedi. It was he, who inspired me to undertake this challenging (at-least for me) task of doing PhD and his was a reassuring presence during all these trying times. He has tolerated my whims and shown great concern for me in his inimitable style.

At the same time, I was fortunate to have Prof. PRK Rao and Prof. PK Chatterjee as my supervisors. Prof. PRK Rao is certainly one of the most outstanding academicians, I have encountered. The supervision of this thesis is only a small part of the overall training that he has imparted to me and his teachings will go a long way in shaping my professional as well as personal conduct in future. Apart from electrical engineering subjects, he introduced me to various fascinating subjects like History and Philosophy of Science. I am greatly impressed by his scholarship, his commitment and sincerity towards teaching and eagerness to learn new things even at his age. I, on my part, was certainly unable to come up to his standards, but what came as a solace to me that as a true teacher, he understood my limitations, displayed a great deal of patience and allowed me to speak freely on all matters. My long hours of marathon discussions with him on academic as well as non-academic matters have considerably sharpened my arguing power.

I recall with pleasure my long association with Prof. PK Chatterjee. He had supervised my masters thesis as well. I am grateful to him for the advice

that he tendered me and the moral strength and support, he provided me throughout the course of this work. I am grateful to him for his affection and concern.

It was a memorable and pleasant experience to live in association with several competent teachers including Profs MU Siddiqi, VP Sinha, RK Bansal, SK Bose and GK Dubey. I learnt my finer points of electrical engineering circuits during my association with Prof. KR Srivathsan as his teaching assistant. He always considered me as a part of his group and was ready to offer any help whenever sought. Several young faculty members of this department treated me like their friends. I am grateful to Drs. AK Raina, G Sharma, SK Bose and Alope Dutta for their affectionate attitude and enquiring about my progress from time to time.

During my stay at IIT, Kanpur, I came in contact with a large number of people and made several friends who have contributed directly or indirectly. I am extremely grateful to Balvinder for his eagerness to help me despite his own busy schedule. He painstakingly read the entire manuscript and offered many constructive suggestions. I thoroughly enjoyed his company and all those 'tea' talks with him particularly during the last phase of this work.

I also enjoyed my association with Venkatesh, Deepak, Vinod, Sandeep, Babu and TSR. They have shared with me the joys and sorrows of doing PhD. I will always remember the useful time spent with them in discussing various topics on this earth. Deepak, Balvinder and Venkatesh have offered every possible help in the typesetting of this thesis. I also wish to acknowledge my several other friends- Arun, Indra, Venkatramani, Shafi, Mangsuli, Ashish, Chaveli, Kanetkar and Manish for their help.

Throughout my stay at Kanpur, the house of Vibha tai and Sardesai jiiji has provided me a comfortable place to relax whenever I wished. They have always treated me like a family member and I am greatly overwhelmed by the affection of their children towards me. I am also thankful to Mrs. Karuna Sharma, Mrs. Mala Chatterjee, Dr. and Mrs. Kulkarni and Charu for their concern.

Finally, it would have been impossible for me to undertake anything of

the kind of doing a PhD without the active support, encouragement and enthusiasm of my parents, Ashish, Anju, Aarti, Girish and Milind. My parents have provided me the much needed emotional support and encouraged my every activity. I have always felt secured in their company. It is not possible to express in words my indebtedness to them. This thesis is dedicated to them.

Abhay Karandikar

# Contents

---

Synopsis	ii
Acknowledgements	ix
List of Figures	xvi
List of Tables	xvii
List of Notations	xix
<b>1 Introduction</b>	<b>1</b>
1.1 Random Multiple Access Problem	1
1.2 Historical Perspective of Random Access Communications	3
1.3 Motivation for the Thesis	10
1.4 Contributions and Organization of the Thesis	11
<b>2 Towards A Unified Framework for Random Access Algorithms</b>	<b>14</b>
2.1 The Common Receiver Model	15
2.2 Characterization of Random Access Algorithm	20
2.2.1 Tsybakov's Definition of Random Access Algorithm	21
2.2.2 Classification of Random Access Algorithms	22
2.2.3 Performance Measures of a Random Access Algorithm	22
2.3 Capacity of Random Access Channel for Infinite User Model	26
2.3.1 Upper Bound on the Capacity	29

2 4	Review of Random Access Algorithms for Infinite User Model	31
2 4 1	ALOHA Algorithm	31
2 4 2	Stable Random Access Algorithms of Class $\mathcal{A}_1^\sim$	34
2 5	Random Access Algorithm for a Finite User Model	43
2 5 1	Definition of Random Access algorithm for Finite User Model	44
2 5 2	Parameters for Characterization of Random Access Algorithm	46
2 5 3	Review of Random Access Algorithms for Finite User Model	48
2 6	Random Access Systems with Multipacket Reception Capability	51
2 6 1	CDMA-RA System Model	52
2 6 2	Review of Random Access Algorithms for Multipacket Reception Channels	54
<b>3</b>	<b>An Optimal Collision Resolution Algorithm for Single Buffer Users</b>	<b>56</b>
3 1	The System Model	57
3 2	Random Access Algorithm for Finite User Model	60
3 3	Dynamics of the System	61
3 3 1	The System State	61
3 3 2	Enumeration of the States	61
3 3 3	Evolution of the System State	62
3 3 4	Evaluation of State Transition Probabilities	64
3 4	The Optimal Collision Resolution Algorithm	67
3 4 1	Statement of the Problem	68
3 4 2	Existence of Optimal Collision Resolution Algorithm	69
3 4 3	Determination of Optimal Collision Resolution Algorithm	70
3 5	Steady State Average Collision Resolution Length	72
<b>4</b>	<b>Suboptimal Collision Resolution Algorithms</b>	<b>80</b>
4 1	General Finite User Model	81
4 1 1	Basic Assumptions	81

4 1 2	Dynamic Evolution of the System	82
4 1 3	The Conditional Probability Distribution of the State	84
4 2	The Optimal Collision Resolution Problem	86
4 3	Suboptimal Collision Resolution Algorithm	87
4 3 1	Suboptimal Collision Resolution Algorithm-1 (SOA1)	87
4 3 2	Suboptimal Collision Resolution Algorithm-2 (SOA2)	89
4 4	Simulation of Suboptimal Collision Resolution Algorithms	90
4 4 1	Procedure for Simulation	91
4 5	Performance of Suboptimal Algorithms in the presence of Feed-back Error	102
4 5 1	Simulation of Suboptimal Algorithms in the presence of Feedback Error	104
<b>5</b>	<b>A Blocked Random Access Algorithm for Buffered Users</b>	<b>106</b>
5 1	Queueing Problem for Buffered Users	107
5 2	System Model	108
5 2 1	Mechanism of Access Protocol	108
5 2 2	The System Markov Chain	111
5 3	Evaluation of the State Transition Probabilities	111
5 3 1	Determination of $\pi_{j,n}^s(k_1)$	113
5 3 2	Determination of $\pi_n^c(k_2)$	116
5 4	Ergodicity of the Markov Chain	118
5 5	Steady State Probabilities	120
5 5 1	Procedure for Computations	120
5 5 2	Numerical Results	121
5 6	Packet Delay Analysis	125
5 6 1	The Delay Components $\mathcal{D}^a$ and $\mathcal{D}_s^d$	127
5 6 2	The Delay $\mathcal{D}_o^d$	129

5 6 3	Numerical Results	134
5 7	Polling Protocol	135
5 7 1	On Comparison of Multiple Access Protocols	135
5 7 2	Description of Polling Protocol	136
5 7 3	Evaluation of State Transition Probabilities	137
5 7 4	Steady State Probability Vector	138
5 7 5	Numerical Results	139
5 7 6	Expected Packet Delay	140
6	An Optimal Collision Resolution Algorithm for Multipacket Reception	145
6 1	The System Model	146
6 2	Random Access Algorithm for CDMA-RA System	148
6 2 1	The System State	149
6 2 2	Dynamic Evolution of the System State	150
6 3	State Transition Probabilities	152
6 4	Determination of Optimal Collision Resolution Algorithm	155
6 4 1	Numerical Results	156
7	Conclusions	163
A	Proof of Existence and Determination of Optimal Algorithm	168
B	Probability of the Tagged Packet Delay	171
	References	173

# List of Figures

---

2 1	Random Access Communication Model	16
2 2	Common Receiver Model	16
2 3	Classification of Random Access Algorithms	23
2 4	Infinite Ternary Tree Graph	28
2 5	Illustration of Window Access Algorithm	36
2 6	Capetanakis's Tree Algorithm	38
4 1	Representation of Finite User Random Access Model	83
4 2	Massey's Model for Feedback Channel	103
5 1	Dynamic Operation of the Protocol	109
5 2	Transmission Epoch	109
5 3	Delay encountered by a Packet	126
5 4	State Transition diagram for TDMA for buffered users	139
6 1	Generalized Common Receiver Model	146

# List of Tables

---

3 1	Optimal Control Values for Possible States for $N = 8$	74
3 2	Optimal Control Values for Possible States for $N = 16$	75
3 3	Expected Collision Resolution Length for $N = 8$	78
3 4	Expected Collision Resolution Length for $N = 16$	79
3 5	Steady State Average Collision Resolution Length for $N = 16$	79
4 1	Active User Patterns	92
4 2	Initial Probability Distribution-1	93
4 3	Simulation Results for Probability Distribution-1	94
4 4	Expected Collision Resolution Length with Known Initial State	94
4 5	Initial State Probabilities-2	95
4 6	Simulation Results for Probability Distribution-2	96
4 7	Initial State Probabilities-3	96
4 8	Simulation Results for Probability Distribution-3	97
4 9	Initial State Probabilities-4	98
4 10	Simulation Results for Probability Distribution-4	98
4 11	Initial State Probabilities-5	99
4 12	Simulation Results for Probability Distribution-5	99
4 13	Initial State Probabilities-6	100
4 14	Simulation Results for Probability Distribution-6	101
4 15	Initial State Probabilities-7	101
4 16	Simulation Results for Probability Distribution-7	102

4 17 Results for Probability Distribution-1 Feedback Error	104
4 18 Results for Probability Distribution-2 Feedback Error	105
5 1 Truncation Size Chosen versus Arrival Probability	122
5 2 Expected Value and Variance of Transmission Epoch Length	122
5 3 Expected Length of Collision Resolution & Successful Packet Transmission Part	124
5 4 Resolution Factor & Throughput	125
5 5 Packet Delays for Buffered Users	135
5 6 Truncation Size vs Arrival Probability	141
5 7 Expected Epoch Length for TDMA protocol	141
5 8 Packet Delays due to Polling Protocol for Buffered Users	144
5 9 Comparison of Packet Delays	144
6 1 Optimal Control for $N = 8$	158
6 2 Optimal Control for $N = 16$	159
6 3 Expected Collision Resolution Length for $N = 16$	162

# List of Notations

---

$\sigma$	Probability of Packet Arrival
$p(X   Y)$	Probability of $X$ conditioned on $Y$
$\Theta(t)$	Feedback history at time $t$
$g^{(x)}(t)$	Transmission history of packet (which arrived at $x$ ) at time $t$
$g_i^j(t)$	Transmission history of $j$ th packet of $i$ th user
$\mathcal{A}^\infty$	Class of algorithms for infinite user model
$\mathcal{A}^N$	Class of algorithms for finite user model
$S(t)$	State of the finite user random access system in a collision resolution epoch at time $t$
$\mathcal{S}$	State Space
$S_i = (N_i, n_i)$	$i$ th element of state space $\mathcal{S}$
$\Phi$	Control policy
$p_{(N_i, n_i), (N_j, n_j)}(u)$	Probability of transition from the ( $N_i, n_i$ ) to ( $N_j, n_j$ ) in an epoch
$\ell_i$	Length of $i$ th epoch

$\pi_{j k}$	Probability of transition from epoch length $j$ to epoch length $k$ during dynamic operation
$L'(N, n)$	Expected length of collision resolution epoch for single buffer or collision resolution part of epoch for buffered users
$L^s(N, n)$	Expected length of succesful packet transmission part of epoch for buffered users
$L_\alpha$	Epoch length under steady state operation
$\binom{n}{r}$	Binomial Coefficient or Number of ways in which $r$ can be chosen out of $n$

# Chapter 1

## Introduction

---

### 1.1 Random Multiple Access Problem

In the last two decades, there has been an explosion of research in the area of random access communication. Beginning with the development of ALOHA in 1970, the use of computer communication networks has become widespread and with that, the interest in the study of random access communication has also intensified. In random access communication, there are many sources of data traffic (called users) trying to send *bursty* messages through a common broadcast communication channel. This system is modelled by what is called the *common receiver model*. This model is discussed in detail in Chapter 2.

In a common receiver model, many bursty transmitters attempt to send messages to a single receiver. The signal received at the receiver may be corrupted by mutual interference due to possible simultaneous transmission of messages. This mutual interference is called collision and usually results in loss of messages. The collided messages thus require retransmissions in order to be received successfully at the common receiver. The random multiple access problem is the efficient scheduling of these messages by the users operating *independently* and in a *decentralized manner* and, the scheduling strategies that the users follow are called *random access algorithms* (RAA) or protocols.

In general, the multiple access problem is concerned with using a common communication channel by many users. This problem has also been considered within the framework of information theory. A good account of this view

(known as multiuser information theory) can be found in the text by Cover and Thomas [9] Gallager [17] has reviewed these two approaches in terms of the underlying communication problems

The key difference between the two approaches is that in random access communication, the emphasis is on the bursty arrival<sup>1</sup> of messages as opposed to uniform regular arrival of messages assumed in multiuser information theory approach. Another important difference is that in random access approach, the channel noise is ignored. The only form of error present on the channel is the destructive interference due to simultaneous transmission of messages. The assumption of ignoring the channel noise in effect, decouples the random multiple access problem from the standard communications problem considered in Shannon's sense.

Although the notion of *bursty* arrival of messages is central to the study of random access system and the term *bursty* has been repeatedly used in the literature, there appears to be no systematic investigations in the characterization of the *burstiness* of a traffic arrival process. The generally understood definition of burstiness is that when the peak arrival rate to the average arrival rate of a user's traffic is high, then that traffic is considered *bursty*. Thus in a multi-user environment, a user may not have anything to transmit most of the time and generates messages in the form of bursts. This situation is encountered in computer networks like local area networks and packet radio networks. Lam [39] introduces the concept of "bursty factor" to characterize a bursty source. Berger [2]<sup>2</sup> assumes that each burst can be encoded in the form of packets which may be either of fixed length or of variable length.

In most of the studies conducted in random access communication, an infinite user model is considered. In this model it is assumed that there are potentially infinite number of users. The total traffic of new packets generated by all users forms a Poisson process. In a finite user model considered in the

---

<sup>1</sup>Some studies consider that a user generates packets while others take the view that packets arrive to the buffer of a user. In this thesis, the terms generation and arrival have been used interchangeably.

<sup>2</sup>See [17] also.

literature, it is assumed that there is a set of finite users and the packet arrival process to each user is assumed to be a discrete time random process<sup>3</sup>. We will discuss this model in greater detail in Chapter 2.

In such a bursty user environment, the fixed assignment multiple access schemes like Time Division Multiple Access (TDMA) and Frequency Division Multiple Access (FDMA) prove highly inefficient. The disadvantage of using these fixed assignment techniques in such situations is that the communication channel is wasted during the period when the assigned user has nothing to transmit while the user having a message to transmit may incur large delays. In such situations, random access algorithms can lead to efficient channel utilization.

The major parameters that are of interest for evaluating the performance of any random access algorithm are throughput, delay and stability of the system. Various definitions have been used in the literature for these parameters [74]. As established in [79], some can be shown to be equivalent definitions, but others are wholly unrelated. These definitions are discussed in Chapter 2. The different definitions have sometimes resulted in confusion leading to wrong comparisons of the protocols. However, broadly speaking, we may define throughput as the expected number of packets successfully transmitted per unit time. The packet delay is defined as the duration from the moment of its generation to the moment of its successful transmission. There are various other interpretations of delay which are discussed in the next chapter. Generally, a random access system is said to be stable for a given arrival rate of packets, if the corresponding expected packet delay is bounded.

## 1.2 Historical Perspective of Random Access Communications

The concern of a communication engineer is to determine the channel capacity or its limit to transmit information reliably and then design communication protocols whose performance may approach this limit. As mentioned

---

<sup>3</sup>Usually this process is a Bernoulli process.

by Massey [43], historically however, communication strategies have been suggested first before the actual modelling and the systematic study of the channel. The study of random access communication also started with the development of ALOHA network in 1970 by Abramson [1]. The objective of this network was to connect computer terminals situated at different locations of the University of Hawaii through a radio link. The development of ALOHA algorithm soon triggered research in random access algorithms. Later on, Capetanakis and Tsibakov suggested stable random access algorithms which perform better than ALOHA algorithm. Subsequently, several algorithms [44, 80, 16, 51, 83] were proposed each claiming better throughput than the previous one. On this explosion of research in suggesting protocols, Massey wrote in [43]

*my far from exhaustive collection of papers on this subject now forms a pile about one meter high, most of which dates from the past five years. There has, however, been scant attention given to channel models for random access communications. This makes for confusion, throughputs are often compared for schemes that in fact require different channel models.*

Although several algorithms were proposed, most of these studies lacked a systematic attempt to develop a general theory of random access algorithms. Tsibakov looked at the structure of the information pattern available to the users and defined a class of random access algorithms by giving a formal notion of a random access algorithm. Pippenger [56] considered the question of maximum achievable throughput for a random access channel and obtained an upper bound on this limit. We briefly look at some of these historical developments in the following.

In the original ALOHA algorithm,<sup>4</sup> if a user has a packet at any instant, it is allowed to transmit this packet. If two or more users attempt to send their packets simultaneously, a collision is said to occur and the packets are destroyed. The unsuccessful users retransmit their packets after waiting for a random amount of time. For the analysis of this algorithm, Abramson introduced the concept of statistical equilibrium similar to Kleinrock's independence assumption made in queueing theory [37]. For a Poisson arrival

---

<sup>4</sup>Also called Pure ALOHA

model of traffic, he assumed that the system ultimately reaches an equilibrium position such that the retransmission traffic adds to the incoming new traffic in such a way that the aggregate traffic remains Poisson. Abramson showed that throughput  $\mathcal{T}$  (i.e., the expected number of packets successfully transmitted per unit time) can be expressed as a function of the total offered traffic  $G$  by the relationship  $\mathcal{T} = Ge^{-2G}$ . Thus the maximum throughput that can be achieved with Pure ALOHA is  $1/2e$  or 0.18 packet/slot.

It was subsequently shown by Roberts [63] that this maximum throughput is doubled, if the channel is divided into time slots and the packets are allowed to begin their transmissions only at the slot beginnings. The length of each slot is assumed to be equal to the packet transmission time. At the end of each slot, the users are informed about the state of the slot, i.e., whether the slot was empty, contained one packet or contained more than one packet. The slot containing one packet is called *success slot* and that with more than one packet is called *collision slot*. This model is called *infinite user ternary feedback* model (or simply infinite user model). We will discuss more about this model in Chapter 2. With these assumptions, the relationship between the throughput and the total offered traffic is given by  $\mathcal{T} = Ge^{-G}$  which shows that the maximum throughput in slotted ALOHA is 0.36 packet/slot.

Two methods of scheduling retransmission of collided packets have been considered in the literature [40]. In a geometrically distributed delay, the collided packet is retransmitted in a slot with some probability. In a uniformly distributed delay, the collided packet selects a time slot for retransmission from some fixed time slots in equally likely manner.

After the introduction of ALOHA algorithm, several attempts were made to rigorously model and analyze the random access channel, with the result that Abramson's assumption of statistical equilibrium was soon found to be erroneous [38, 19]. Markov chain formulation of the slotted ALOHA channel showed that the algorithm is unstable for the infinite user model. Kleinrock and Lam [38] have discussed the stability of the ALOHA algorithm. Their simulation results suggest that the channel becomes unstable independently of the arrival rate of packets if the number of users is very large. The instability

was said to occur when the number of blocked packets (*i.e.*, those waiting for retransmission) becomes arbitrarily large. For a finite user model, it was established that the slotted ALOHA algorithm can be either stable or unstable depending upon the number of users and the traffic arrival rate [33].

Various control schemes [34, 62, 28, 72, 67] have been suggested to improve the stability of the ALOHA algorithm. These control schemes are essentially based on two policies:

- Making the retransmission probability dependent on the state of the system
- Limiting the access of newly generated packets to the network

The former is called the retransmission control policy [41, 19] and the latter, the input control policy [49].

Note that the ALOHA algorithm, even with control schemes, can give a maximum throughput of 0.36 packet/slot only. During the early seventies, it was not clear whether algorithms with higher throughput can be developed. A major breakthrough in the study of random access algorithms was, however, provided by Capetanakis [7] and independently by Tsibakov [81]. They showed that their algorithms not only give higher throughput than ALOHA, but are inherently stable. Capetanakis' algorithm is known as the Binary Tree algorithm while Tsibakov introduced the same algorithm using the interpretation of a stack operation. This algorithm is therefore called Stack algorithm.

The key concept behind Capetanakis's tree algorithm is that, unlike in slotted ALOHA with control algorithm, it is possible for the collided users to make use of the past history of feedback outputs and schedule their packets in a cooperative manner so as to retransmit them successfully [44]. Such strategies are also called *Collision Resolution Algorithms*. The original tree algorithm is based upon the idea of tossing a fair coin and dividing the collided users into two parts. Those users who flip *head* retransmit in the very next slot and those who flip *tail* wait till any possible collision of the first group has been resolved after which they transmit in the next slot. No new packets are transmitted

till the initial contending users have resolved their collisions

Massey [44] proposed a modification to this algorithm to remove certain redundant operations performed by the algorithm. The algorithm was later improved by Gallager [16] who has given a Markov chain analysis for the infinite user model [18]. This algorithm was independently proposed and analyzed by Tsybakov and Mikhailov in [80]. Tsybakov's analysis is more powerful in the sense that it readily adapts to situations where users may toss a biased coin. It is also elegant as it provides recursive expressions for calculating the throughput of the algorithm.

The original tree algorithm of Capetanakis assumed *blocked channel access* for its operation, where the new packets are blocked till the colliding packets have been resolved. This algorithm therefore calls for the continuous monitoring of channel by the packets. Tsybakov and Vvedenskaya [81] considered a free access stack algorithm in which, unlike in the blocked access tree algorithm, a newly generated packet does not wait for contending users to resolve their collisions but is immediately transmitted in the next slot. It then joins the group of the contending users if unable to transmit successfully in that slot. Thus the algorithm requires that a packet monitor the state of the channel only in the period from the moment of its initial transmission till the moment of its successful transmission. Tsybakov showed that such an algorithm is indeed stable and achieves a throughput of 0.384 packet/slot. It was subsequently recognized [22] that such algorithms belong to a class of algorithms known as the *Limited Sensing Algorithms*.

For the infinite user model, Massey's modified tree algorithm gives a throughput of 0.375 packet/slot and 0.462 packet/slot for blocked channel access and window channel access respectively while Gallager's algorithm gives a throughput of 0.48 packet/slot.

The packet delay analysis of these stack/tree algorithms has proved to be an extremely formidable task. Various attempts [7, 44, 14, 20, 31] have been made to perform the delay analysis. Most of these studies, however, obtain only upper and lower bounds on the expected packet delay.

It was Pippenger [56] who first addressed the question of maximum achiev-

able throughput for a random access channel. This maximum possible throughput is called capacity of the random access channel. This brought in the notion of an optimal random access algorithm. He showed that the capacity of an infinite user (ternary feedback) model of random access channel cannot be greater than 0.744 packet/slot.

The bound on the maximum throughput was further tightened by Molle [50] and, Cruz and Hajek [10] to 0.673 and 0.612 respectively. The best known upper bound due to Tsibakov and Likhonov [77] is 0.568. Vvedenskaya and Pinsker [83] have considered a class of random access algorithms known as first come first served (FCFS) algorithms. They have derived an optimal algorithm which achieves capacity on the subclass of these FCFS algorithms.

The key observation made by Pippenger [56] is that there is an uncertainty about the location of packets on the generation time axis and the objective of any random access algorithm or protocol is to resolve this uncertainty in order to transmit packets successfully. The protocol can gain one bit of more information by way of collisions and empty slots than by successful slots, meaning thereby that the protocol gains more information from failures than from success. The bound on the maximum throughput reflects the trade-off between the twin objectives of transmitting a packet successfully on the one hand and gaining information by way of collisions and empty slots (in order to transmit packets successfully at later stages) on the other hand. Another important conclusion of Pippenger's argument is that if the collision multiplicity is known at the end of each slot, then throughput arbitrarily close to one can be achieved. Panwar [55] has shown recently that it is indeed possible to derive such an optimal random access algorithm which in the limit can achieve a throughput of one.

Apart from the ternary feedback model, random access systems with binary feedback have also been considered [3]. Generally, three types of binary feedback have been considered. They are collision/no-collision, something/nothing and success/failure feedbacks. Berger and Mehrvarı [3] have discussed the random access algorithms for these models.

For a finite population model, Capetanakis has considered the tree algorithm

with deterministic addressing [6]. When the probability of each user having a packet is known, then Capetanakis's proposal yields an optimum tree protocol. This optimum tree protocol (called Dynamic Tree Algorithm in [6]) can be shown to be a generalization of TDMA.

The group testing algorithms considered in statistics [69] were also proposed for the finite user random access model [85]. The group testing algorithms which perform better than binary tree search algorithm of Capetanakis have been studied for both ternary and binary feedback models [4].

In all the above studies, it has been assumed that if more than one packet is transmitted in a slot, then collision occurs and all the packets are completely destroyed. Recently a more general model has been considered, wherein the simultaneous transmission of two or more packets does not necessarily result in the destruction of all packets. There are many such random access channels such as code division multiple access (CDMA) [60], packet radio network with capture effects [8] and multichannel networks [86]. The first such study in the context of CDMA was made by Raychaudhuri [60]. This study suggests that appropriate use of multiaccess coding can provide normalized throughput-delay characteristics better than that of ALOHA. A recursive retransmission control strategy for slow frequency hopped-random access system was suggested by Hajek [27]. One advantage of CDMA is that it can be used in asynchronous environment. The throughput in asynchronous CDMA random access system with ALOHA has been investigated by several authors [52, 12, 35, 87]. Ghez *et al* [24, 25] have studied a general model for an infinite user random access channel with multipacket reception capability.

Mehravari [47] has investigated collision resolution algorithms for a multipacket reception model where if more than  $d$  packets (where  $d > 2$ ) are transmitted, they are destroyed but the simultaneous transmission of  $d$  or fewer packets result in success. We will review some of these results in the next chapter. Collision resolution algorithms for random access CDMA systems were also investigated by Hu and Chang [30].

Other studies made in random access communications are those of multiple access algorithms like Carrier Sense Multiple Access (CSMA), CSMA with

Collision Detection (CSMA/CD) and reservation protocols. These and several other algorithms for packet radio networks have been discussed in the text by Bertsekas and Gallager [18].

### 1.3 Motivation for the Thesis

A large number of random access algorithms have been proposed in the literature. Most of these algorithms have been studied for an infinite user model. Some of the pioneering contributions in this area have been made by Capetanakis [7], Gallager [16], Tsybakov and his group [81, 80], Pippenger [56] and Molle [50]. Pippenger's seminal paper has given an insight into the nature of operation of a protocol, while Tsybakov's formal definition of a random access algorithm showed that many previously proposed algorithms can be studied within this definition. Tsybakov and his group have provided some useful research directions towards formulating the optimal algorithms [74]. Despite all these significant contributions, there are still some open questions in the analysis of random access algorithms and the issue of optimality of these algorithms is far from settled.

On the other hand, the finite user population model has not been studied to that extent. After Capetanakis's paper [6], Berger *et al* [4] have made some useful progress by proposing the application of group testing algorithms in random access context.

This provides us the motivation to take up the study of optimal random access algorithm for a finite user model. The objective of this thesis is to investigate whether a systematic formulation of optimal random access algorithm can be given. The motivation behind this is that such a formulation is not only desirable for the completeness of the theory but also can provide a basis for comparisons of different algorithms and establish their interrelationship.

Sometimes, it may not be possible to derive an optimal algorithm but the approach adopted and the difficulty encountered thereby in obtaining an optimal algorithm may give us some clues to suggest good suboptimal algorithms. Indeed as will be shown in Chapters 3 and 4, we have derived,

under certain assumptions, an optimal collision resolution algorithm for a class of random access models. When these assumptions are relaxed, we show the difficulty in obtaining an optimal algorithm and suggest two suboptimal algorithms. Our claims about these algorithms are substantiated by simulations.

The steady state analysis under dynamic operation of this algorithm for buffered users is then carried out. Finally, the algorithm is extended to channels with multipacket reception capability. This extension is motivated by the fact that such channels can model coded random access systems whose practical usefulness in packet radio networks has been demonstrated recently [82].

We now summarize the important contributions and organization of the thesis in the following section.

## 1.4 Contributions and Organization of the Thesis

In Chapter 2, we give a systematic treatment of some random access algorithms within Tsibakov's formulation. We consider here random access systems which can be modelled by the common receiver model. We discuss this model in detail. The definition of a random access algorithm for an infinite user model provides a generic framework for the study of these algorithms. We show how various random access algorithms can be classified using this definition. We then discuss various performance measures like delay, throughput, stability, backlog and average collision resolution lengths considered in the literature. The issues related to the capacity of a random access channel and an optimal algorithm are reviewed. We then discuss the operations of some random access algorithms for the infinite user model. The finite user model is considered next. Tsibakov's definition of random access algorithm for the infinite user model enables us to define a random access algorithm for the finite user model. We provide a classification of random access algorithms according to this definition and then consider our attention to a particular class of these algorithms. We briefly review some of the existing algorithms which can be considered to be members of this class. Finally, we consider random access systems with

multipacket reception capability. A class of such systems can be shown to be a generalization of the common receiver model. We therefore consider this generalized model and provide a brief sketch of the recent progress made in the study of random access algorithms for this model.

In Chapter 3, we introduce the assumptions made for our finite user random access system. We consider a class of random access algorithms for this model known as blocked random access algorithms. The blocked random access algorithms are characterized by a collision resolution algorithm with blocked channel access. We consider the problem of an optimal collision resolution algorithm for this model. To begin with, we assume in this chapter that the number of active users at the beginning of a collision resolution epoch is known. We also assume that the users can transmit only single packet per epoch. We identify an appropriate state of the system and show that the problem of optimal collision resolution algorithm is equivalent to the first passage problem of a Markovian Decision Process. The dynamic programming technique is then applied to derive an optimal collision resolution policy. The algorithm is illustrated with numerical examples. Finally, we present the steady state operation of our system using this collision resolution algorithm and suggest an approximate way to calculate the average collision resolution length.

In Chapter 4, we remove the restriction of the number of active users at the beginning to be known and consider a more general model. Thus instead of the state being known exactly, we assume that the probability distribution over the initial states is given. The conditional probability vector can be updated using the past control action and the feedback observed at each stage. It is shown that this conditional probability vector constitutes the appropriate new state and the problem of optimal collision resolution can be considered within the framework of Partially Observable Markovian Decision process. The solution of this problem is, however, computationally tedious. Hence, instead of determining the optimal algorithm, we suggest two suboptimal algorithms. One is based upon choosing the control corresponding to the state which has the maximum value of probability, while the other is based upon choosing the control corresponding to the expected state as if these were the actual states.

The performance of these algorithms is studied by simulations. The expected collision resolution lengths obtained from simulations for different probability distribution due to the two suboptimal algorithms is then compared with bounds on the optimum performance. The degradation in performance due to the two suboptimal algorithms is not very significant for most cases. We then discuss the performance of these algorithms in the presence of feedback error by simulations.

In Chapter 5, we consider the case of users with infinite buffers. For such buffered users, we consider an access protocol which employs the optimal collision resolution algorithm of Chapter 3 for the case of single buffer. We perform the steady state analysis of this protocol. The sequence of transmission epoch lengths is shown to be a Markov chain. The expressions for evaluating the state transition probabilities, steady state probabilities and the expected transmission epoch lengths are derived. The average packet delay analysis is then performed. Numerical results for different arrival probabilities for a fixed set of users are presented. We then consider a type of polling protocol and perform a similar analysis for this protocol. The two protocols are then compared.

In Chapter 6, we consider random access systems with multipacket reception capabilities. We consider a generalized common receiver model with transmitter based code assignment. Each transmitter is provided with a set of orthogonal codes. The number of such codes is much smaller than the total number of users in the system. Each user, before transmitting its packet selects one of these codes as its address and then encodes its packet with this address. If two or more users choose the same code, then their packets collide. We extend the optimal collision resolution algorithm of Chapter 3 to such systems. The collision resolution algorithm specifies the number of users to be enabled for each code. The dynamic programming equations are solved to derive an optimal algorithm. The algorithm is illustrated with numerical examples. In Chapter 7, we summarize the important conclusions of this thesis and indicate some directions for future investigations.

## Chapter 2

# Towards A Unified Framework for Random Access Algorithms

---

In this chapter, we examine some of the issues involved in the study of random access algorithms and attempt an unified treatment of these algorithms within Tsybakov's formulation. It is not the aim of this chapter to give an exhaustive review of all the algorithms, but the presentation has been made here with a view to illustrate the basic features of random access communication. The objective here is to point out, in a generic way, the nature of problems being considered in the literature.

We begin this chapter by discussing the Common Receiver Model. We then give Massey's qualitative definition of a random access algorithm. The formal definition of a random access algorithm proposed by Tsybakov for an infinite user model is next introduced. It is then shown how various random access algorithms can be classified using this definition. The various performance measures used to characterize a random access algorithm are then discussed. Pippenger [56] using some insightful arguments has obtained an upper bound on the maximum possible throughput for a class of random access models. We examine the issue of optimality of an algorithm and the capacity of a random access channel in the light of Pippenger's arguments. The mechanism of operation and performance analysis of some representative random access algorithms are then reviewed.

We next consider the finite user model. Tsybakov's definition of a random

access algorithm for an infinite user model provides us the motivation to define a random access algorithm for the finite user model. Accordingly, we introduce in this chapter, the formal definition of a random access algorithm for the finite user model. We consider a class of such random access algorithms and review some of the existing algorithms that can be included within this class.

Finally, we extend our discussion to random access systems with multipacket reception capability. Due to the presence of other parameters, there are various possibilities of modelling such systems. We restrict our attention to the model which can be considered to be a generalization of our common receiver model. We conclude this chapter by reviewing some random access algorithms for this generalized model.

## 2.1 The Common Receiver Model

The model used for the study of random access communication is called the Common Receiver Model. In this model, there are many bursty sources of data traffic (users) which send messages to a single receiver through a common communication channel. In random access communication model, as shown in Figure 2.1, each user is equipped with a transmitter (Tx) and a receiver (Rx). If we focus our attention on the receiver of a particular user and consider an instance, when other users try to send their messages to this receiver, then this situation is equivalent to the common receiver model. Thus the common receiver model is an instance of a random access communication model. After taking propagation delay into account, the random access communication model can be considered as the superposition of several such common receiver models. For our purposes, it suffices to focus on the common receiver model.

The messages are assumed to be broken up into segments called packets. These packets may be of either fixed length or variable length. We assume that fixed length packets are transmitted by the users. The packets may also be encoded with some kind of error-correcting code. The other assumptions

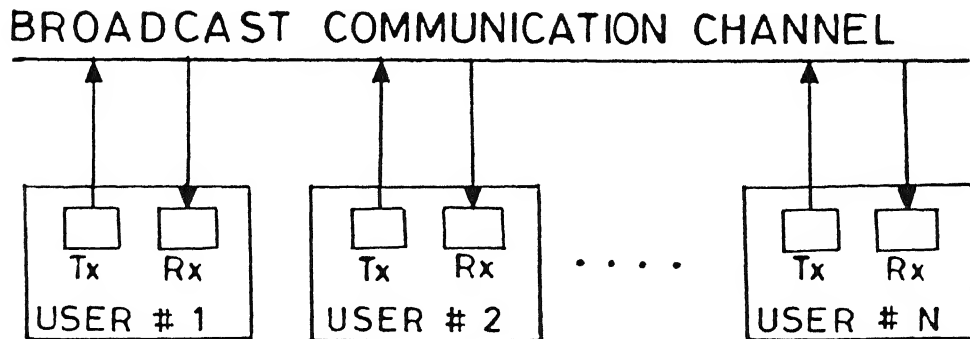


Figure 2.1 Random Access Communication Model

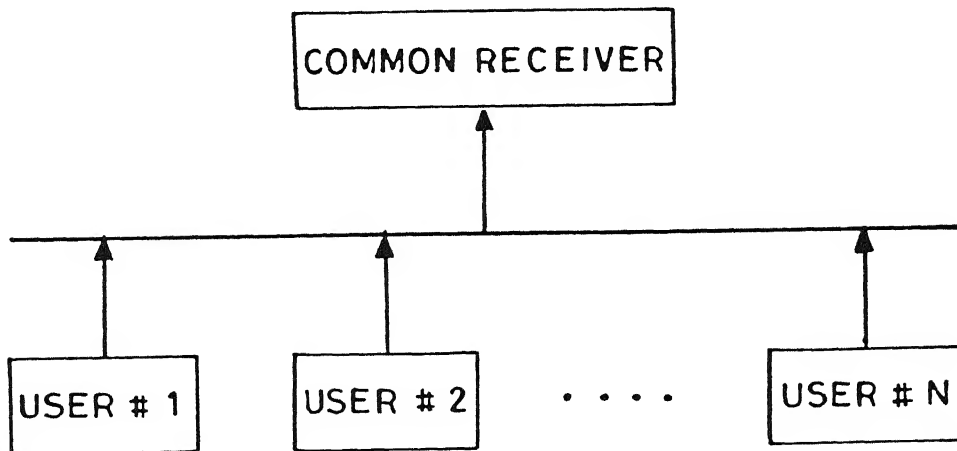


Figure 2.2 Common Receiver Model

about this model are as follows

- 1 It is assumed that the channel time is divided into short time intervals called slots. The users can transmit their packets only at the beginning of a slot. The length of each slot is assumed to be equal to the time taken for the transmission of a packet.
- 2 The transmission of two or more packets in a slot results in collision or the destruction of all the transmitted packets. If only one packet is transmitted in a slot, then that packet is received successfully. In other words, it is assumed that the only form of noise present on the channel is the destructive interference due to simultaneous transmission of packets.
- 3 It is assumed that at the end of each slot, all users know about the state of the slot through feedback information. The feedback assumes ternary values 0, 1 or 2 depending upon, whether the slot was empty or contained one packet (success) or contained more than one packet (collision). The common receiver broadcasts this feedback on a separate feedback channel.
- 4 The feedback is assumed to be immediate and errorless and thus represents the true state of the slot. It is also assumed, as in most studies, that all users are aware of the past feedback values regardless of whether they had packets for transmission.

Apart from the above basic assumptions, two models of user population are considered in the literature. These are *Finite User Model* and *Infinite User Model*.

In the *finite user model*, [6, 4] there is a fixed, finite set of users. Each user generates packets independent of each other. The packet generation process at each user is usually assumed to be a discrete time Bernoulli process. In the *infinite user model*, there are potentially infinite number of users. The aggregate traffic of new packets by all the users is usually assumed to be a Poisson process.

At this point, it is appropriate to make the following comments on these assumptions

**Remark 2.1.1** The assumption of slotted system requires that all users are synchronized at the slot level. This synchronization at the slot level may not be an easy task. The random access algorithms for unslotted system have also been considered in the literature. In unslotted system, the users can transmit their packets at any instant. The assumption of slotted system, however, facilitates the analysis. Accordingly, we restrict our attention to slotted systems only.

**Remark 2.1.2** The assumption that the feedback is immediately available at the end of the slot is somewhat restrictive. It essentially ignores the effect of propagation delays. Such assumption is usually justified when the round trip propagation time is much smaller than one slot unit duration. Such a condition however fails in Satellite Networks. We may note here that Capetanakis [44] has suggested a random access algorithm for such satellite channels.

**Remark 2.1.3** The absence of channel noise both on the actual random access channel and the feedback channel is the most crucial assumption which decouples the random access problem from Shannon's communication problem. Some authors [42] have studied the performance of random access algorithms in the presence of channel noise.

**Remark 2.1.4** All the above remarks also apply to the corresponding assumptions made in subsequent chapters of this investigation.

**Remark 2.1.5** The communication channel with assumptions 1 to 5 is referred to as the *random access channel* in the sequel. Similarly the common receiver model with infinite user Poisson arrivals is called the infinite user model and that with a finite population of users is called the finite user model.

**Remark 2.1.6** There have been some departures from the above assumptions. We describe the following important deviations from these assumptions.

- 1 *Binary Feedback* Instead of assuming that the users receive ternary valued feedback information, some studies [3, 4] have assumed binary feedback. Three types of binary feedbacks have been considered in the literature
  - (a) *conflict/no-conflict* This feedback informs the users whether the slot contains more than one packet (conflict) or, equal to one packet or no packet (no-conflict)
  - (b) *something/nothing* This feedback informs the users whether the slot contains one or more packet (*ie*, something) or, it contains no packet (*ie*, nothing)
  - (c) *success/failure* This feedback informs whether the slot contains only one packet (*ie*, success) or, no packet or more than one packet (*ie*, failure)

The something/nothing feedback and success/failure feedback characterize public key cryptosystems and spread spectrum random access systems respectively

- 2 *D-ary Feedback* A general  $D$ -ary feedback model has also been considered. This feedback informs the users whether 0, 1, 2, ...,  $D$  packets are transmitted or more than  $D$  packets are transmitted in a slot. Thus the feedback informs the users about the collision multiplicity up to a certain degree. This model has been studied by Tsybakov [73], Georgiadis and Papantoni-Kazakos [21] and Shiv Panwar [55]
- 3 *Limited Sensing* We have assumed that the users monitor the channel continuously whether or not they have packets for transmission. Some models, however, have been considered which do not require a user to observe the state of the channel continuously, but only from the moment it has a packet ready for transmission till the moment the packet has been transmitted successfully. The algorithms for such models are called limited sensing algorithm. This model was first considered by Tsybakov [81] whose stack algorithm did not require the restriction of continuous sensing for its operation. Later on, Georgiadis *et al* [22] and Humblet [32] have discussed limited sensing algorithms

**Remark 2.1.7** Recently there is a lot of interest in the study of random access algorithms for channels with multipacket reception capability [47, 25]. In such systems, it is assumed that even if more than one packet is transmitted in a slot, then some packets may still be received successfully. The effect of capture phenomenon in packet radio networks [8] and code division multiple access-random access system [60] can be modelled by such channels. We will discuss this model in Section 2.6.

## 2.2 Characterization of Random Access Algorithm

In this section, we examine how to characterize a random access algorithm. We first give the qualitative definition of a random access algorithm due to Massey [44].

**Definition 2.1** *Random Access Algorithm is a strategy that the users follow to access a (random access) channel.*

Random access algorithm may employ a collision resolution algorithm and a channel access algorithm. These are defined as follows.

**Definition 2.2** *A collision resolution algorithm is a strategy for the (transmission and) retransmission of packets by the users with the property that after each collision, all packets involved in the collision are eventually retransmitted successfully and all users eventually and simultaneously become aware that these packets have been successfully transmitted.*

**Definition 2.3** *A channel access algorithm is a rule by which users decide the slot for the first time transmission of a newly generated packet.*

We may note here that according to above definition, ALOHA is not a collision resolution algorithm. Although Massey's definition summarizes the basic characteristic of a collision resolution algorithm, it does not tell whether such an algorithm exists at all and if it does, then how it can be synthesized.

Tsybakov and Mikhailov [80] gave a mathematical definition of random access algorithm for an infinite user model. We restrict our attention to the infinite user model in this section. We will return to the finite user model in Section 2.5.

### 2.2.1 Tsybakov's Definition of Random Access Algorithm

Following [74], in a slotted random access system, the packets are transmitted only at times  $t \in I$  where  $I = \{0, 1, \dots\}$ . Let  $N$  be the number of users in the system. For the finite user model  $N < \infty$ , while for the infinite user model  $N = \infty$ . Let at the instant  $t$ , the feedback vector  $\Theta(t) = \{\theta(1), \theta(2), \dots, \theta(t)\}$  where  $\theta(r)$  denotes the feedback observed at the time  $r$  for  $r = 1, 2, \dots, t$ . This feedback may assume values of 0, 1 or 2 for empty slot, successful slot and collision slot respectively.

Let  $x$  denote the time of generation of a packet, then  $x \in R^+$ , where  $R^+$  is the semi-infinite continuous time axis, i.e.,  $R^+ = [0, \infty)$ . These instants of packet generations form a Poisson process for an infinite user model. Let  $g^{(x)}(t) = \{g^{(x)}(1), g^{(x)}(2), \dots, g^{(x)}(t)\}$  where for  $r = 1, 2, \dots, t$  we have

$$g^{(x)}(r) = \begin{cases} 0, & \text{if the packet generated at time } x \\ & \text{was not transmitted during slot } (r-1, r) \\ 1, & \text{if the packet generated at time } x \\ & \text{was transmitted during slot } (r-1, r) \end{cases}$$

Then for an infinite user model, a random access algorithm can be defined as follows.

**Definition 2.4** *Random access algorithm is an algorithm which evaluates the probability function  $f[x, \Theta(t), g^{(x)}(t)]$ , with which a packet generated at time  $x$  will be transmitted at the instant  $t$ , i.e., in the slot  $(t, t+1)$ .*

**Definition 2.5** *Random access algorithm is a causal algorithm if  $f[x, \Theta(t), g^{(x)}(t)] = 0$  for  $x > t$ .*

We restrict our attention to causal algorithms only.

### 2.2.2 Classification of Random Access Algorithms

Tsybakov's definition of random access algorithm provides some interesting classification of these algorithms [74]. Let us denote the class of all random access algorithms by  $\mathcal{A}^\infty$ . We discuss below various subclasses of  $\mathcal{A}^\infty$ .

- 1 If the function  $f[x, \Theta(t), g^{(x)}(t)] = \alpha \forall x, \Theta(t)$  and  $g^{(x)}(t)$  and where  $\alpha$  is a specified probability taking values from 0 to 1, then the random access algorithm that results is the well known slotted ALOHA algorithm. Let us denote this class by  $\mathcal{A}_0^\infty$ .
- 2 Consider a random access algorithm where the function  $f[x, \Theta(t), g^{(x)}(t)]$  is such that it assumes binary values of either 0 or 1. Let the class of all such algorithms be denoted by  $\mathcal{A}_1^\infty$ . For an algorithm from  $\mathcal{A}_1^\infty$ , we have for a packet generated at time  $x$

$$g^{(x)}(t+1) = f[x, \Theta(t), g^{(x)}(t)]$$

The above relation easily follows from the definition of  $g^{(x)}(t)$ . Since  $g^{(x)}(t)$  can be recursively computed, the class  $\mathcal{A}_1^\infty$  is characterized by random access algorithms which (with a little abuse of notation) can be denoted by  $f[x, \Theta(t)]$ . This class includes Tree algorithm [7], Stack algorithm [81] and Gallager's interval searching algorithm [16]. These algorithms are discussed in Section 2.4. If the function  $f[x, \Theta(t)] = 1$  for  $x \in (t-1, t]$ , then we have free access stack algorithm of Tsybakov [81].

- 3 Another subclass of  $\mathcal{A}^\infty$  is the class consisting of all algorithms with function  $f[x, \Theta(t), g^{(x)}(t)]$  depending only on  $\Theta(t)$  and not on  $x$  and  $g^{(x)}(t)$ . An example of such class is the adaptive algorithm [74].

The classification of all these algorithms is depicted in Figure 2.3.

### 2.2.3 Performance Measures of a Random Access Algorithm

After having defined a random access algorithm, we turn our attention to various performance measures which are used to characterize the efficiency

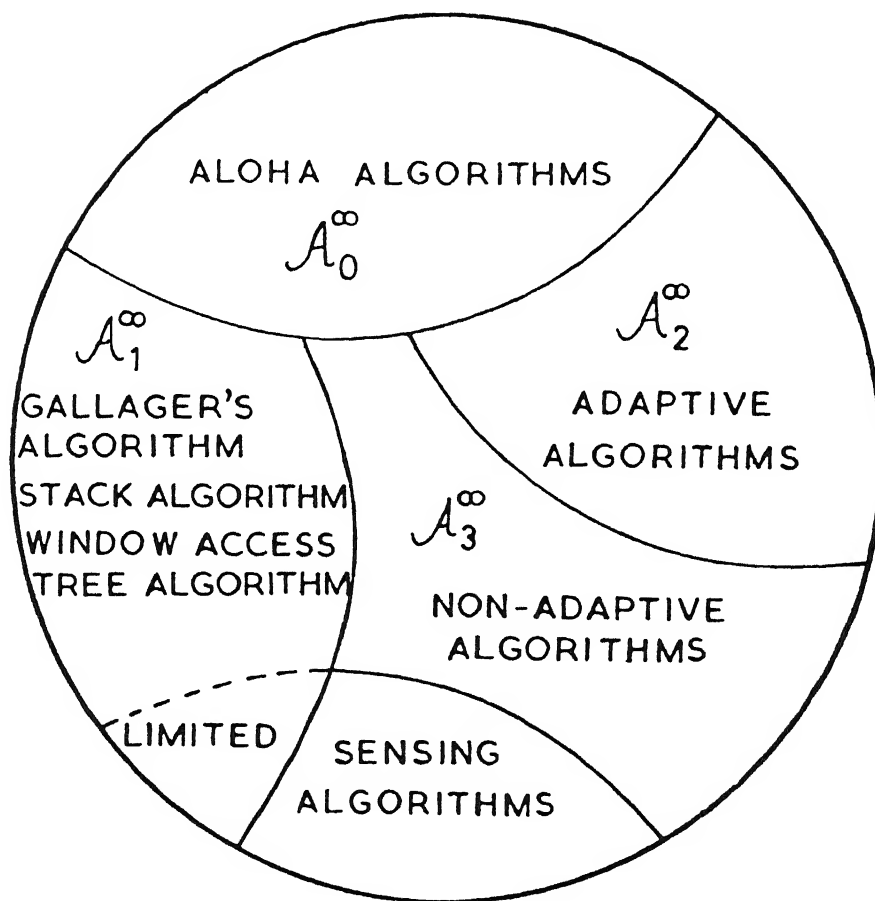


Figure 2.3 Classification of Random Access Algorithms

of a random access algorithm. Two main criteria for evaluating a random access algorithm are delay and throughput. Steady state analysis under dynamic operation of the algorithm involves the study of stability behavior of the algorithm. Some other parameters of interest are backlog of the system and average collision resolution length. We summarize the notion of delay, throughput, stability, backlog and average collision resolution length in the following

### Packet Delay

The packet delay is the time from the moment of a packet's generation to the moment of its successful transmission. The average packet delay has been interpreted by various authors in the following ways

- 1 Let  $\delta_x$  be the delay encountered by a packet generated at time  $x$ . Then the average packet delay can be defined [74] as

$$\mathcal{D}_1 = \lim_{x \rightarrow \infty} E[\delta_x] \quad (2.1)$$

- 2 Another definition of packet delay used in the literature [20] is as follows. Let the packets be indexed as 1, 2, ... according to the order of their arrival instants. Let  $\delta_j$  be the delay experienced by the  $j$ th packet. Then the average packet delay is defined as

$$\mathcal{D}_2 = \lim_{m \rightarrow \infty} E \left[ \frac{1}{m} \sum_{j=1}^m \delta_j \right] \quad (2.2)$$

- 3 For an infinite user model employing Gallager's algorithm, the average packet delay is defined in [31] in the following way:

$$\mathcal{D}_3 = \lim_{i \rightarrow \infty} E[\delta^i] \quad (2.3)$$

where  $\delta^i$  is the delay encountered by a randomly chosen packet arriving in the  $i$ th enabled interval<sup>1</sup>

---

<sup>1</sup>The enabled interval is defined in Section 2.3

## Throughput

There are various definitions of throughput considered in the literature

- 1 According to one definition of throughput, it is the supremum of the input traffic arrival rates  $\lambda$  such that the packet delay remains bounded. Thus throughput  $\mathcal{T}_1$  can be written as

$$\mathcal{T}_1 = \sup\{\lambda \mid \mathcal{D} < \infty\} \quad (2.4)$$

where  $\mathcal{D}$  is any of the delays defined earlier

- 2 Let  $\tilde{t}$  be the time instant when all packets generated before time  $t$  have been successfully transmitted. Then throughput can be defined as

$$\mathcal{T}_2 = \lim_{t \rightarrow \infty} \left[ \frac{t}{E[\tilde{t}]} \right] \quad (2.5)$$

Here the arrival rate  $\lambda$  is assumed to be equal to one

- 3 Let  $n(t)$  be the number of packets successfully transmitted from 0 to  $t$ . Then let  $\mathcal{T}_3(\lambda)$  be defined as follows

$$\mathcal{T}_3(\lambda) = \begin{cases} \lim_{t \rightarrow \infty} E \left[ \frac{n(t)}{t} \right] & \text{if } \mathcal{D} < \infty \\ 0 & \text{otherwise} \end{cases} \quad (2.6)$$

The throughput of the random access algorithm,  $\mathcal{T}_3$  is then defined as

$$\mathcal{T}_3 = \sup_{\lambda} \mathcal{T}_3(\lambda) \quad (2.7)$$

- 4 The following definition of throughput is a much stronger definition than that given by (2.4)

$$\mathcal{T}_4 = \sup\{\lambda \mid \lim_{k \rightarrow \infty} \lim_{j \rightarrow \infty} \text{Prob}[\delta_j < k] = 1\} \quad (2.8)$$

## Stability

A random access algorithm is inherently stable if the throughput  $\mathcal{T} > 0$  and unstable if  $\mathcal{T} = 0$ . Note that ALOHA algorithm is inherently unstable for the infinite user model. Random access algorithms employing collision resolution by tree or stack algorithm, however, are stable algorithms.

### Backlog

A packet is said to be backlogged at any instant, if it has not been transmitted successfully till that instant. Let  $N_t$  be the number of backlogged packets at time  $t$ , then the average backlog of the system is defined as

$$\mathcal{N} = \lim_{t \rightarrow \infty} E[N_t] \quad (2.9)$$

The stability of the random access system may also be defined with respect to this backlog. Thus a random access system is said to be stable if the following condition is satisfied

$$\lim_{k \rightarrow \infty} \lim_{t \rightarrow \infty} \text{Prob}[N_t < k] = 1 \quad (2.10)$$

### Average Collision Resolution Length

As defined earlier, let  $\tilde{t}$  be the time instant when all packets generated before time  $t$  have been successfully transmitted. Then Tsybakov [74] defines the average delay associated with an algorithm  $f$  as follows

$$\mathcal{D}(f) = \lim_{t \rightarrow \infty} E[\tilde{t} - t] \quad (2.11)$$

Under steady state condition, as it will become evident from Section 2.4, this delay is the same as the expected collision resolution length in stack algorithms.

## 2.3 Capacity of Random Access Channel for Infinite User Model

**Definition 2.6** Let the capacity of a random access channel, with random access algorithms of class  $\mathcal{A}^\infty$ , be denoted by  $C(\mathcal{A}^\infty)$ . Then  $C(\mathcal{A}^\infty)$  is given by

$$C(\mathcal{A}^\infty) = \sup\{T \mid f \in \mathcal{A}^\infty\} \quad (2.12)$$

where  $T$  is one of the throughputs defined earlier.

**Definition 2.7** A random access algorithm with throughput  $T$  equal to the capacity  $C(\mathcal{A}^\infty)$  is called an Optimal Algorithm.

The determination of the capacity of a random access channel for the infinite user model over class  $\mathcal{A}^\infty$  is still an open problem. Attempts have been made to formulate the capacity problem over class  $\mathcal{A}_1^\infty$  algorithms. However even for this class, only bounds are available. Vvedenskaya and Pinsker [83] have recently considered a subclass of  $\mathcal{A}_1^\infty$  and obtained an optimal algorithm for the infinite user model over this subclass.

### Formulation of the Capacity Problem

Since the bounds on the capacity of a random access channel are available only over the class  $\mathcal{A}_1^\infty$  algorithms, we restrict our attention to random access algorithms over class  $\mathcal{A}_1^\infty$  in this section. The class  $\mathcal{A}_1^\infty$  is characterised by the function  $f[x, \Theta(t)]$  which assumes the values of either 0 or 1. The problem of determining the capacity for the infinite user model can be reduced to the following problem.

Let us consider the set  $B[t] = \{x \mid f[x, \Theta(t)] = 1, x \in R^+\}$ . Thus set  $B(t)$  at time instant  $t$  contains all such generation instants  $x \in R^+$  for which packets are transmitted. Thus any random access algorithm from class  $\mathcal{A}_1^\infty$  can be specified by the collection of measurable sets  $\{B(t)\}$ . The packets are generated according to Poisson process of intensity  $\lambda = 1$ . We say that the set  $B(t)$  is enabled at time  $t$ , if all those packets with their generating instants belonging to this set are allowed to transmit in the slot  $(t, t + 1)$ .

Any algorithm in class  $\mathcal{A}_1^\infty$  can be considered as embedded in an infinite ternary tree graph as shown in Figure 2.4. Each node is connected by three branches to three other nodes. These three branches correspond to whether the slot is empty, contains one packet (i.e., success) or contains more than one packet (i.e., collision). For a given value of  $\Theta(t) = \{\theta(1), \theta(2), \dots, \theta(t)\}$ , there is a unique path through this infinite tree. This path will have  $t$  branches and  $t + 1$  nodes. The measurable sets associated with these nodes are  $B(1), B(2), \dots, B(t)$ .

Now let us consider the finite subset  $\mathcal{E} = [0, x] \subset R^+$ . Let  $K$  denote the set of all nodes of the tree representing the algorithm. Let  $k \in K$  be some

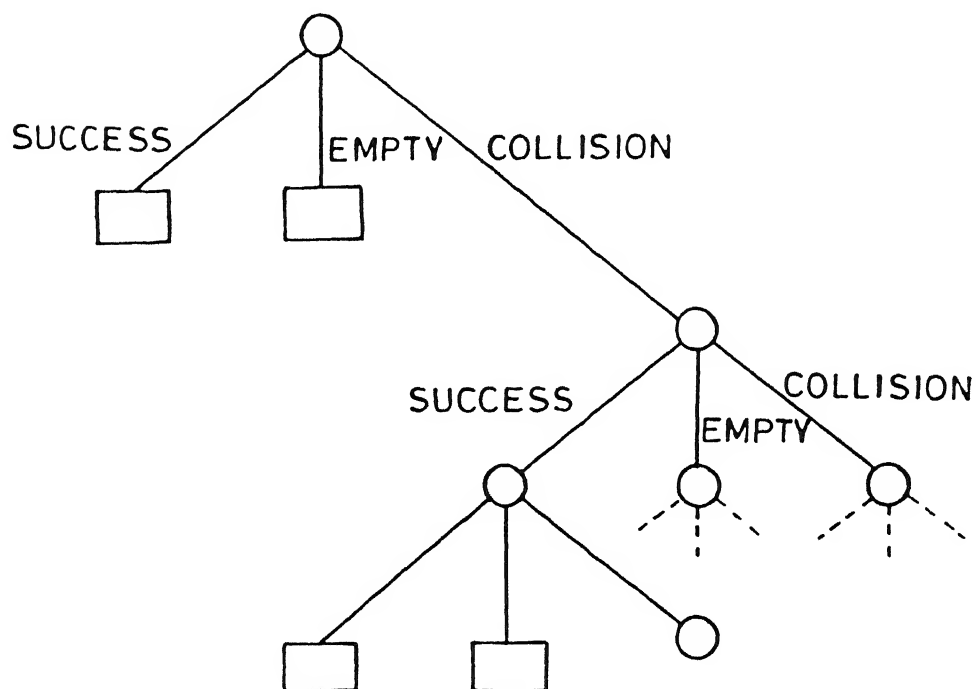


Figure 2.4 Infinite Ternary Tree Graph Nodes are indicated by circles with the root at the top, leaves or terminal nodes are indicated by the squares

node. Let  $B_k$  denote the measurable set associated with  $k$ <sup>2</sup>. Let  $\rho_k$  denote the probability that the random access algorithm used for transmitting packets generated within the interval  $\mathcal{E}$  will ever visit  $k$ . Then the average number of slots required for the transmission of all packets of  $\mathcal{E}$  is given by

$$E[\tilde{t}(x)] = \sum_{k \in K} \rho_k \quad (2.13)$$

Note that  $\tilde{t}(x)$  is the time instant, when all packets generated before the instant  $x$  have been successfully transmitted. Let  $q_{0,k}$ ,  $q_{1,k}$  and  $q_{2,k}$  be the probabilities that empty, success or collision is observed, when the set  $B_k$  is enabled at node  $k$ . Then the average number of packets transmitted is  $\sum_{k \in K} \rho_k q_{1,k}$ . We then have

$$\sum_{k \in K} \rho_k q_{1,k} = x \quad (2.14)$$

The right hand side follows from the fact that the intensity of Poisson process is assumed to be one here.

The capacity of the random access channel with respect to throughput  $\mathcal{T}_2$  can then be determined by the relation:

$$\begin{aligned} \mathcal{C}(\mathcal{A}_1^\infty) &= \sup_{f \in \mathcal{A}_1^\infty} \left\{ \lim_{x \rightarrow \infty} \frac{x}{E[\tilde{t}(x)]} \right\} \\ &= \sup_{f \in \mathcal{A}_1^\infty} \left\{ \lim_{x \rightarrow \infty} \frac{\sum_{k \in K} \rho_k q_{1,k}}{\sum_{k \in K} \rho_k} \right\} \end{aligned} \quad (2.15)$$

No general method is available for evaluating this expression of the capacity of random access channel. However many authors have derived an upper bound on this capacity. We present below some of these results.

### 2.3.1 Upper Bound on the Capacity

Using information theoretic arguments, Pippenger has shown that the capacity over class  $\mathcal{A}_1^\infty$  is upper bounded by 0.7448. Following [2], we present below a sketch of Pippenger's approach.

---

<sup>2</sup>This set  $B_k$  is equal to  $B(t)$  if the node  $k$  was visited at the instant  $t$  and so on.

As explained above, when an algorithm over class  $\mathcal{A}_1^\infty$  is used to transmit all packets belonging to some finite interval  $[0, x]$  on the generation time axis, then there is an associated path through the infinite tree. When all packets of the interval  $[0, x]$  have been transmitted successfully, say at  $\tilde{t}(x)$ , we say that the interval  $[0, x]$  has been resolved. At this stage we reach the terminal node. Let the random variable  $L$  denote this terminal node. The entropy of  $L$ , then equals the average amount of information required to specify which path through the infinite tree was followed to reach the terminal node from the root node. It can be shown that  $H(L)$ , the entropy of  $L$  is bounded from above in the following way

$$H(L) \leq E[\tilde{t}(x)] H\left(\frac{x}{E[\tilde{t}(x)]}\right) + \{E[\tilde{t}(x)] - x\} \log 2 \quad (2.16)$$

where  $H(y) = -y \log y - (1 - y) \log(1 - y)$  and  $E[\tilde{t}(x)] = \sum_{k \in K} \rho_k$ . Similarly it can be shown [2] that

$$H(L) \geq x \log e \quad (2.17)$$

From Equations 2.16 and 2.17, we have

$$x \log e \leq H(L) \leq E[\tilde{t}(x)] H\left(\frac{x}{E[\tilde{t}(x)]}\right) + \{E[\tilde{t}(x)] - x\} \log 2 \quad (2.18)$$

As we take the limit of this expression as  $x \rightarrow \infty$ ,  $x/E[\tilde{t}(x)]$  gives the throughput  $\mathcal{T}$  (with  $\mathcal{T} = \mathcal{T}_2$ ) of the random access channel. Taking therefore the limit of this equation as  $x$  tends to infinity, we can write

$$\mathcal{T} \log e \leq H(\mathcal{T}) + (1 - \mathcal{T}) \log 2 \quad (2.19)$$

The capacity  $\mathcal{C}$  is the maximum value of  $\mathcal{T}$  satisfying this equation. It can be shown that this maximum value of  $\mathcal{T}$  is 0.7448. Thus

$$\mathcal{C} \leq 0.7448 \quad (2.20)$$

**Remark 2.3.1** In this section, we have defined the capacity with respect to throughput  $\mathcal{T}_2$ . An important theorem due to Tsybakov [79] relates the throughput  $\mathcal{T}_2$  to  $\mathcal{T}_1$ . This theorem states that if an algorithm  $f$  has a throughput

$\mathcal{T}_2 > 0$  and  $E[\tilde{T}^2] < \infty$  where  $\tilde{T}$  is the time instant when all packets generated before time  $t$  have been successfully transmitted, then there exists an algorithm  $f^*$  with throughput  $\mathcal{T}_1 = \mathcal{T}_2$

The bound on the capacity was further sharpened by Molle [50] using a completely different argument to 0.673. Later on, Cruz and Hajek [10] refined Molle's arguments and reduced the upper bound to 0.612. Tsybakov and Mikhailov [48] obtained an upper bound of 0.578. The best known bound of 0.568 is due to Tsybakov and Likhonov [77].

## 2.4 Review of Random Access Algorithms for Infinite User Model

In this section, we review some of the existing algorithms proposed in the literature for an infinite user model. Historically, the earliest and simplest random access algorithm was ALOHA. There are various versions of ALOHA and all can be grouped under the class  $\mathcal{A}_0^\infty$ . But these algorithms are inherently unstable and require some control mechanism. On the other hand, algorithms over  $\mathcal{A}_1^\infty$  are not only stable but provide higher throughput. Our interest in  $\mathcal{A}_1^\infty$  is also motivated by the appealing conjecture of Tsybakov [74] that an optimal algorithm over the class  $\mathcal{A}_1^\infty$  is also optimal over the entire class  $\mathcal{A}^\infty$ . In order to develop an appreciation of the importance of algorithms over  $\mathcal{A}_1^\infty$ , we first discuss ALOHA algorithm.

### 2.4.1 ALOHA Algorithm

In a slotted ALOHA protocol considered here, a newly generated packet is transmitted in the first slot after its generation. If two or more packets are transmitted in a slot, they collide and become backlogged. The backlogged packets are transmitted with fixed probability  $\alpha$  in each successive slot until they are successfully transmitted<sup>3</sup>. We now show that such a random access

---

<sup>3</sup>In another version of ALOHA, the newly generated packet, instead of transmitting immediately, may transmit in the next slot with probability  $\alpha$  or with some different probability  $\alpha'$ .

system is unstable

### Instability of ALOHA

Let  $N_t$ ,  $t \in I$  be the number of backlogged packets at time  $t$ . Let  $\lambda$  be the intensity of the overall traffic. Let  $X_t$  denote the number of newly generated packets transmitted during the slot  $(t, t + 1)$ . Let  $Y_t$  denote the number of backlogged packets transmitted during the slot  $(t, t + 1)$ .  $\{X_t, t \in I\}$  is assumed to be a sequence of independent and identically distributed random variables. In our infinite user model, this is a Poisson process, i.e.,

$$c_i = \text{Prob}[X_t = i] = \frac{e^{-\lambda} \lambda^i}{i!} \quad (2.21)$$

Since  $\alpha$  is the probability with which backlogged packets retransmit, we have

$$b_j(n) = \text{Prob}[Y_t = j \mid N_t = n] = \binom{n}{j} \alpha^j (1 - \alpha)^{n-j} \quad (2.22)$$

It can be shown that  $\{N_t, t \in I\}$  is a Markov chain. According to the definition of stability, our system is stable if  $\lim_{i \rightarrow \infty} \text{Prob}[N_t < j] = 0$  for all finite values of  $j$ . To prove that this condition does not hold for ALOHA algorithm being considered here, it is sufficient to show that the Markov chain  $\{N_t, t \in I\}$  is not ergodic. Indeed it has been shown [19] that the Markov chain  $\{N_t, t \in I\}$  is not ergodic for  $\lambda > 0$ .

### Control of ALOHA

The instability of ALOHA has generated several interesting control problems. It can be shown [19, 28, 72, 62] that ALOHA algorithm can be made stable by the closed loop control policies. One such class of policies is based upon choosing the retransmission probability on the basis of the number of backlogged users. Thus the retransmission probability is no longer constant but is a function of the number of backlogged users. It can be shown [19] that if a control policy chooses the retransmission probability  $\alpha(t)$  during the slot  $(t, t + 1)$  based on  $N_t$ , then the system is stable if  $\lambda < d$  and unstable if  $\lambda > d$ .

where  $d = \lim_{n \rightarrow \infty} [c_1 b_0(n) + c_0 b_1(n)]$ . It follows from this that the optimum policy which yields the maximum throughput will choose the optimum retransmission probability  $\alpha^*$  by the relation  $\alpha^* = 1/n$  where  $n$  is the number of backlogged users. For  $\alpha = \alpha^*$ , the optimum value of  $d$  is  $d = [1 - 1/n]^{n-1}$  and the maximum throughput obtained is  $1/e = 0.368$  packet/slot.

Note however that the number of backlogged users is not generally known to the users. The information available to the users is the feedback information  $\Theta(t)$ , the history of packet transmissions  $g^{(x)}(t)$  and the time of packet arrival  $\tau$ . Segall [67] has obtained exact recursive equations for estimating the number of backlogged users given the past feedback values. For an infinite user model, however, these estimation equations are infinite dimensional. Therefore it is necessary to consider suboptimal control policies. One such policy suggested by Hajek and van Loon [28] makes use of the feedback information to give a recursive updating law of the retransmission probability. Our ALOHA algorithm can now be represented by  $f[\Theta(t)] = \alpha(t)$ . A recursive control of the following form was suggested in [28]<sup>4</sup>

$$\alpha(t+1) = G[\alpha(t), \theta(t)]$$

where  $G$  is a function  $G : [0, 1] \times \{0, 1, 2\} \rightarrow [0, 1]$

It was shown in [28] that with this type of control policy, stability can be achieved for arrival rate  $\lambda < 1/e = 0.368$ .

Another suboptimal control policy which is similar to *certainty equivalent control* was suggested by Rivest [62]. This is also known as the Pseudo Bayesian strategy. We know that the optimum value of retransmission probability at time  $t$  is  $\alpha = \alpha^* = 1/n$  where  $n$  is the value of the number of backlogged users at time  $t$ , i.e.,  $N_t = n$ . In Rivest's scheme, we use an estimate of the number of backlogged users. Let  $\hat{N}_t$  denote an estimate of the number of backlogged users at time  $t$ . This estimate is updated according to the following rule

$$\hat{N}_{t+1} = \begin{cases} \max\{1, \hat{N}_{t+1} - 1 + \hat{\lambda}\} & \text{if the slot } (t, t+1) \text{ is empty} \\ \hat{N}_t + \frac{1}{e-2} + \hat{\lambda} & \text{if the slot } (t, t+1) \text{ is successful or collision} \end{cases}$$

where  $\hat{\lambda}$  is an estimate of the arrival rate  $\lambda$ . The above updating law is

---

<sup>4</sup>A more general policy can be of the form expressed by  $\alpha(t+1) = G[\alpha(0), \alpha(1), \dots, \alpha(t), \Theta(t)]$

motivated by an approximation of the exact Bayesian formula for updating the optimal estimate of  $N_i$  given the past feedback observations. It was shown by Tsitsiklis [72] that such a scheme is stable for  $\lambda < 1/e$ . An estimate of  $\lambda$  can be obtained on line but it was proved in [72] that if  $\hat{\lambda} = 1/e$  is chosen to update the estimate of the number of backlogged users, the algorithm achieves the same throughput which would have been obtained for the case where  $\lambda$  is known.

We may note here that even the *controlled* ALOHA algorithms use only the global feedback information available to all users, *i.e.*, the feedback vector. Thus all users use the same retransmission probability. These algorithms do not make use of the transmission history  $g^{(x)}(t)$  and the generation time of packets. Capetanakis's tree algorithms, Tsybakov's stack algorithm and Gallager's algorithm seek to incorporate these informations. Since these algorithms make use of the available information in a better way, they can be expected to yield higher throughput. Indeed as we shall see below that these algorithms not only give higher throughput but are also inherently stable. All these algorithms form a class denoted by  $\mathcal{A}_1^\infty$ .

### 2.4.2 Stable Random Access Algorithms of Class $\mathcal{A}_1^\infty$

In this Section, we discuss some of the random access algorithms which can be included in class  $\mathcal{A}_1^\infty$ . These random access algorithms can be shown to consist of a channel access algorithm and a collision resolution algorithm. As already mentioned in Section 2.2, collision resolution algorithms are used for resolving collisions and channel access algorithms specify when packets may join the collision algorithm. We first discuss the channel access algorithms.

#### Channel Access algorithms

Mainly three channel access algorithms have been considered in the literature. They are

- Blocked Access

- Window Access
- Free Access

**Blocked Access** In blocked access, the operation of the random access system can be divided into several transmission intervals called epochs. The packets generated in one epoch are transmitted for the first time in first slot of the next epoch. If a collision occurs in this slot, *i.e.*, if more than one packet had been generated in the previous epoch, the collision resolution algorithm resolves this collision. No new packet can transmit in this epoch, while the collision resolution algorithm is in operation.

**Window Access** The window access is similar to the blocked access in the sense that in window access also, the packets that arrived during a specific period are transmitted in an epoch, while the other packets wait till the end of this epoch. The window access algorithm operates on the principle of decoupling the transmission time axis from the packet arrival time axis. The packet arrival time axis can also be divided into several epochs. Then in window access algorithm, packets that arrived during the  $i$ th arrival epoch are transmitted in the first *utilizable* slot [44] following the collision resolution epoch for packets that arrived during  $(i - 1)$ th arrival epoch. Here the term *utilizable* slot is used, because the collision resolution epoch for packets arrived during  $(i - 1)$ th epoch may end before the  $i$ th arrival epoch. Since we have considered constant window sizes in this access algorithm, some slots may be wasted if the collision resolution epoch ends earlier<sup>5</sup>. The window access algorithm has been illustrated in Figure 2.5.

In blocked and window access algorithms, the function  $f[x, \Theta(t)] = 0$ , if the packet arrival time  $x$  does not belong to the previous epoch and the specified window.

**Free Access** In free access algorithm, a newly arriving packet does not wait but transmits immediately in the next slot of its arrival. If a collision

---

<sup>5</sup>This problem can be removed by using variable window sizes.

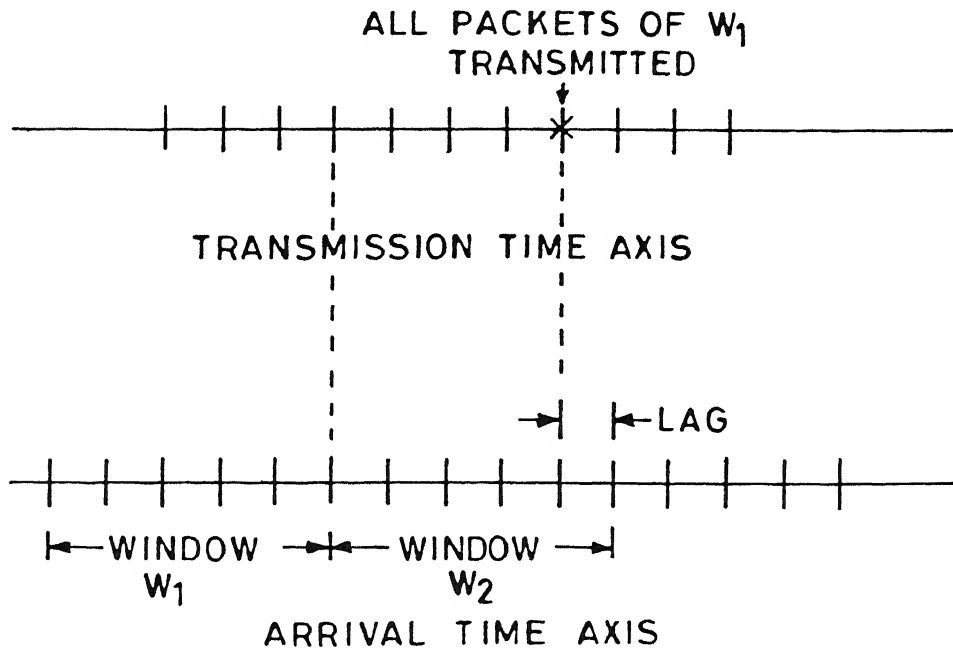


Figure 2.5 Illustration of Window Access Algorithm

occurs in the slot, then the packet joins the group of other collided packets and follow the collision resolution algorithm. The operation of free access algorithm can be best explained by Tsybakov's stack algorithm and will be discussed later in this section.

### Collision Resolution Algorithm

The first collision resolution algorithm was proposed by Capetanakis [7] and independently by Tsybakov [78]. Tsybakov's stack interpretation of the collision resolution algorithm [81] lends naturally to its use with free access. Later on Gallager [16] proposed another collision resolution algorithm for the infinite user model. We discuss below the operation of some of these algorithms.

1. *Capetanakis Tree Algorithm* The tree collision resolution algorithm of Capetanakis can be used in conjunction with either blocked access or window access. The algorithm can be explained as follows—

All users involved in collision divide themselves into two parts. The algorithm first operates on one of these parts. The users belonging to this part are allowed to transmit in the next slot. If a success or an empty feedback is observed, then the collision of this part is resolved and the algorithm then operates on the second part. However, if collision occurs due to transmission of the first part, then this part is further subdivided into two parts. This procedure continues till all users have resolved their collisions. The algorithm then returns to the initial second part and repeats the procedure. The operation of the algorithm can be best illustrated with help of binary tree as shown in Figure 2.6. The root node indicates the first slot of the collision resolution epoch. The splitting of users involved in collision in this slot generates two subtrees from the root. The users can split themselves in two ways–

- By tossing a fair coin. Those tossing *head* join the left subtree and transmit in the very next slot, while those tossing *tail* join the right subtree and wait till the users on the left subtree have resolved their collisions.
- By using the arrival time of packets. The channel access algorithm specifies an initial time interval. After collision, all packets that arrived during the first half of this interval join the left subtree and retransmit in the very next slot, while packets during the second half of the interval join the right subtree.

We note from above that the operation of Capetanakis's tree algorithm does not distinguish between success and empty slots. It was, however, Massey [44] who noted that if a collision slot is followed by an empty slot, then it means that the set of users who tossed *head* is empty and therefore outcome of the next slot due to transmission of the other set of users is predetermined to be a collision. The users therefore, can immediately toss their coins and continue with the collision resolution algorithm. This modified algorithm yields higher throughput than the basic tree algorithm. While Capetanakis's tree algorithm with blocked access and window access gives throughputs of 0.346 and 0.429 respectively, the

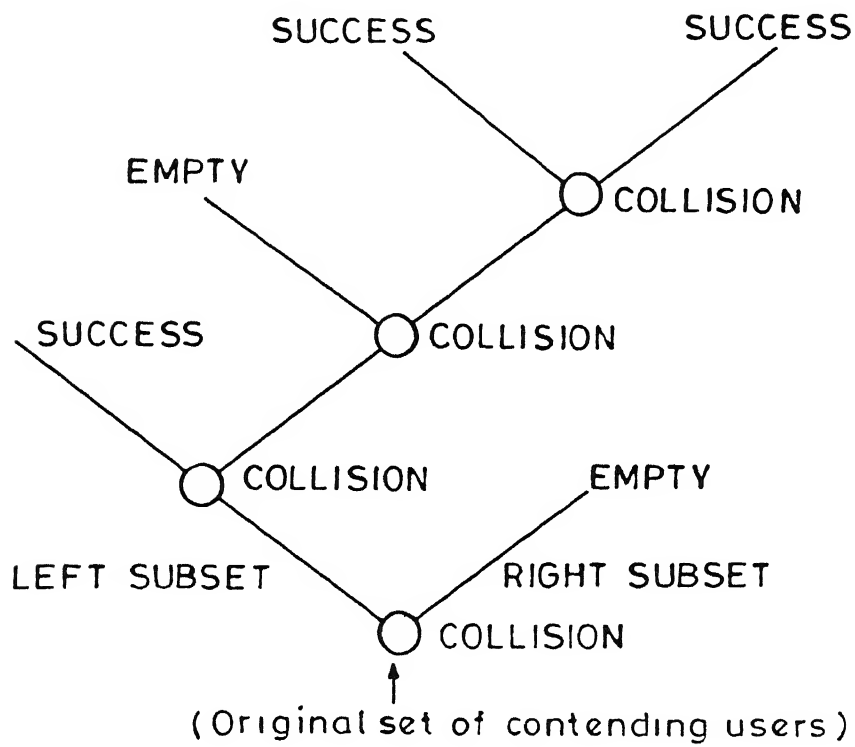


Figure 2.6 Capetanakis's Tree Algorithm

modified tree algorithm gives throughput of 0.375 and 0.462 with the respective channel access algorithms

The modified tree algorithm, however, is extremely sensitive to feedback errors. The algorithm can enter into a deadlock if due to feedback error an empty slot at any stage is mistaken to be a collision slot by the users. The algorithm will cause the splitting of users "involved in collision" into two parts. But since there were no such users, this will result in empty slot. The modified algorithm, however, will again cause splitting as this empty slot is preceded by a collision slot. Thus the algorithm continues splitting, resulting in a series of empty slots. The problem can be resolved if, after a series of  $R$  empty slots is received, the algorithm allows the set of users to transmit in the next slot without splitting. The value of  $R$  can be chosen appropriately depending on the feedback error and the desired performance.

We now provide below a gist of the throughput analysis of tree algorithm with blocked access for an infinite user model.

The number of slots, a collision resolution epoch lasts is called collision resolution length. Let  $L_n$  denote the expected collision resolution length when the number of users transmitted in the initial slot of the epoch is  $n$ . Then from the description of Capetanakis's tree algorithm, we can write the following relations

$$L_0 = L_1 = 1 \quad (2.23)$$

$$L_n = 1 + \sum_{j=0}^n (L_j + L_{n-j}) \binom{n}{j} \alpha^j (1-\alpha)^{n-j} \quad (2.24)$$

Here we have assumed that the users toss a biased coin for splitting themselves. With probability  $\alpha$ , the users join the left subtree and with probability  $1 - \alpha$ , they join the right subtree.

For a fair coin  $\alpha = 1/2$  and we can write for  $L_n$

$$L_n = \frac{1 + 2 \sum_{j=0}^{n-1} L_j \binom{n}{j} 2^{-n}}{1 - 2^{1-n}} \quad (2.25)$$

Massey [44] has given tight upper and lower bounds on the expected collision resolution lengths

$$2.8810N - 1 \leq L_n \leq 2.8867N - 1 \quad \text{for } n \geq 4 \quad (2.26)$$

Let  $\{\ell_i, i = 1, 2, \dots\}$  denote the sequence of collision resolution lengths. Similarly let  $\{\eta_i, i = 1, 2, \dots\}$  denote the sequence of number of packets transmitted in the first slot of collision resolution epochs. Then we have

$$Prob[\eta_{i+1} = n \mid \ell_i = \ell] = \frac{(\lambda\ell)^n e^{-\lambda\ell}}{n!} \quad \text{for } n = 0, 1, 2, \quad (2.27)$$

It can then be shown that [44]

$$E[\eta_{i+1}] = \lambda E[\ell_i] \quad (2.28)$$

Since  $L_n = E[\ell_i \mid \eta_i = n]$ , using this equation and the bounds from Equation 2.26 on  $L_n$  it can be deduced that

$$E[\eta_i] \leq \frac{\lambda}{1 - \gamma\lambda} [1 - (\gamma\lambda)^i] \quad \forall i \geq 0 \quad (2.29)$$

where  $\gamma = 2.8867$ . If we take the limit of this expression as  $i \rightarrow \infty$  and if  $\lambda < 1/\gamma$ , then

$$\lim_{i \rightarrow \infty} E[\eta_i] = E[\eta] \leq \frac{\lambda}{1 - \gamma\lambda} \quad (2.30)$$

Thus the expected backlog is finite if  $\lambda < 0.3465$ . Similarly it can be shown that the expected backlog is infinite if  $\lambda > 0.3471$ .

2. *Tsybakov's Free Access Stack Algorithm* The tree algorithm was originally proposed to operate with blocked access. Simultaneously and independently, Tsybakov proposed blocked access stack algorithm. Both the algorithms are similar, except in the representation of their operations. However, the stack interpretation easily allows the use of collision resolution algorithm with free access or non blocked channel access algorithms. The free access stack algorithm was proposed by Tsybakov [81]. While in the blocked access algorithms, the function defining the complete random access algorithm, i.e.,  $f[x, \Theta(t)]$  depends upon the entire vector

$\Theta(t) = \{\theta(1), \theta(2), \dots, \theta(t)\}$ , in free access stack algorithm  $f[r, \Theta(t)]$  does not depend on  $\theta(i)$  for  $i < r + 1$ . Thus the user need to monitor the channel only when there is a packet for transmission. The operation of the basic stack algorithm has been summarized in [81]

The analysis of the stack algorithm is more difficult to perform. The collision resolution epoch in stack algorithm is the time interval which begins with a collision slot and ends at the moment, when all users know for the first time that not only the packets involved in the initial collision, but also those packets that arrived during this interval have been successfully transmitted.

As before, let  $L_n$  denote the expected collision resolution length with  $n$  packets being transmitted in the first slot of the epoch. The asymptotic analysis of Fayolle *et al* [14] and Mathys *et al* [45] indicates that the  $\lim_{n \rightarrow \infty} L_n/n$  does not converge but exhibits an oscillating behavior. Mathys [45] showed that the stable throughput that can be achieved for users tossing fair coins is 0.36 packet/slot.

- 3 *Interval Searching Algorithm*: The interval searching algorithm was proposed by Gallager [16], Tsybakov and Mikhailov [80] and Ruget [65] simultaneously for an infinite user model. This algorithm is similar to the tree algorithm, but the splitting of users into two parts is done using arrival time of the packets. The algorithm starts by enabling the time interval  $[0, x]$ . If it results in the slot being empty or success, then the collision resolution epoch ends and the algorithm then selects the next interval. However, if a collision occurs, then the enabled time interval is split into two parts and packets in the left half of the interval retransmit in the next slot. Gallager [16], [18] observed that if a collision slot is followed by another collision then no information is obtained about the number of packets in the right half of the interval. Then this right half can be added back to the arrival time axis. Thus the collision resolution epoch in an interval searching algorithm that begins with a collision slot, ends after two successive successful slots have been observed. At this stage, a portion of the initially enabled interval on the Poisson arrival

axis would have been resolved in the sense that all those packets that arrived during this portion have been transmitted successfully by the end of the collision resolution epoch

The interval to be enabled initially at the beginning of the collision epoch can be optimized for maximum throughput. The optimum interval searching algorithm has a throughput of 0.4888. Thus this algorithm yields the highest throughput amongst all known algorithms for the infinite user Poisson arrival model. The interval searching algorithm has the additional property that all packets are transmitted in the order of their arrival. It is also referred to as the First Come First Served (FCFS) algorithm.

Let us define the following quantities

$W(x)$  = expected length of the collision resolution epoch conditioned on the event that the collision resolution algorithm enabled an interval of length  $x$

$T(x)$  = expected length of fraction of the resolved interval conditioned on the event that the collision resolution algorithm enabled an interval of length  $x$  at the beginning

Let  $W_n(x)$  and  $T_n(x)$  denote respectively the values of  $W(x)$  and  $T(x)$  conditioned on the event that the initial collision multiplicity is  $n$ . Then we can write

$$W(x) = \sum_{n=0}^{\infty} W_n(x) \frac{e^{-x} x^n}{n!} \quad (2.31)$$

$$T(x) = \sum_{n=0}^{\infty} T_n(x) \frac{e^{-x} x^n}{n!} \quad (2.32)$$

The quantities  $W_n(x)$  and  $T_n(x)$  can be calculated recursively as shown in [80]. If we now use the following definition of throughput which is already given in Section 2.2.3

$$\mathcal{T} = \sup\{\lambda \mid \lim_{k \rightarrow \infty} \lim_{j \rightarrow \infty} \text{Prob}[\delta_j < k] = 1\}$$

then Tsybakov [80] has shown that the throughput  $\mathcal{T}$  as defined above can be calculated by the following expression

$$\mathcal{T} = \max_x \frac{x T(x)}{W(x)} \quad (2.33)$$

which can be shown to yield  $\mathcal{T} = 0.487$

- 4 Limited Sensing Algorithms Limited sensing algorithm is a generalization of the concept of free access stack algorithm. We may mention here that free access stack algorithms achieve lower throughput than that of the blocked access or window access stack algorithms for the infinite user model.

An advantageous feature of free access is that the users need not know the past channel feedback history. But at the same time this puts a newly generated packet into asynchronism with the system in the sense that a user is unable to decide whether a collision resolution algorithm is in process or not. It was Humblet [32] who observed that a user after generating a packet can start monitoring the feedback for a long time until it is able to detect the boundaries of collision resolution epoch. It then transmits its packet and participates in the collision resolution process. Georgiadis *et al.* [23] exploited this idea and proposed modification to Gallager's interval searching algorithm. They showed that the throughput of the algorithm depends upon a system parameter and indeed it is possible for the algorithm to achieve the throughput of 0.487 as the value of this system parameter increases.

## 2.5 Random Access Algorithm for a Finite User Model

We have discussed above several random access algorithms for an infinite user model. We know that the random access algorithm for an infinite user model can be described by the function  $f[x, \Theta(t), g^{(x)}(t)]$  which depends upon the packet's arrival time, the feedback history and packet's own transmission

history In this section, we define a random access algorithm for the finite user model

In a finite user model, there are large number of finite users and all these users are indexed or addressed Thus each user is identified by its address In general, the users may be provided with buffers of finite or infinite storage capacity The packet arrival process to each user is assumed to be discrete time stochastic process It is usually Bernoulli process but the packets may also arrive according to Poisson or any other arrival process These packets are stored in the buffer in the order of their arrival, till they are transmitted successfully If the buffer is of finite capacity then the packets which are generated after the buffer is full, are assumed to be lost The random access algorithm governs the transmission of the packets which are stored in the users' buffers

Conceptually, we may also view this problem as a queueing system where each user has a queue of packets to be transmitted The channel acts as the server However, the server has no knowledge of which users have packets and how many of these packets are present in a user's buffer The information about packet queues is thus distributed

### 2.5.1 Definition of Random Access algorithm for Finite User Model

The random access algorithm for finite users may thus depend upon the address of the user, the time of packet arrival and the feedback history Let  $N$  be the total number of users in the system Let these users be addressed as  $1, 2, \dots, N$  The packets arriving to a user may also be indexed as  $j = 1, 2, \dots$  in the order of their arrival in buffers Let  $\Theta(t) = \{\theta(1), \theta(2), \dots, \theta(t)\}$ , where  $\theta(r)$  is the feedback received at the instant  $r$  Similarly let  $g_i^j(t)$  denote the history of transmission of the packet indexed as  $j$  in  $i$ th user where  $i \in \{1, 2, \dots, N\}$  Then  $g_i^j(t) = \{g_i^j(1), g_i^j(2), \dots, g_i^j(t)\}$ , where  $g_i^j(r) = 1$  if the  $j$ th packet of the  $i$ th user was transmitted during the slot  $(r - 1, r)$  and  $g_i^j(r) = 0$  otherwise, for  $r = 1, 2, \dots, t$  Tsybakov's definition of random access algorithm for an infinite user model has been discussed in Section 2.2.1 Motivated by this definition,

we now provide the definition of random access algorithm for a finite user model

**Definition 2.8** *The random access algorithm for a finite user model is an algorithm which specifies the probability function  $f[i, j, \Theta(t), g_i^j(t)]$  with which the packet indexed as  $j$  in  $i$ th user will be transmitted in the slot  $(t, t + 1)$*

We restrict our attention to only those class of random access algorithms which allow the first packet of a user's buffer to be transmitted successfully first and till then, other packets of that user wait. After this first packet has been transmitted successfully, all other packets advance their positions in the buffer by one. In other words, we consider only those algorithms for which  $f[i, j, \Theta(t), g_i^j(t)] = 0$  if  $j$ th packet is not the first packet of  $i$ th user's buffer at time  $t$ . Let us denote the class of all such random access algorithms by  $\mathcal{A}^N$

**Definition 2.9** *The class  $\mathcal{A}^N$  comprises all those algorithms for which the function  $f[i, j, \Theta(t), g_i^j(t)] = 0, \forall j \in \{1, 2, 3, \dots\}$  if the following condition is not satisfied*

$$\begin{aligned} f[i, k, \Theta(\tau), g_i^k(\tau)] &= 1 \\ \text{and } \theta(\tau + 1) &= 1 \end{aligned}$$

for some  $\tau \in \{0, 1, \dots, t - 1\}$  and  $\forall 1 \leq k \leq j - 1$

Along the lines of Section 2.2.2, we can now classify random access algorithms for a finite user model. Two important subclasses of the class  $\mathcal{A}^N$  are as follows

1. **Class  $\mathcal{A}_0^N$**  If the function  $f[i, j, \Theta(t), g_i^j(t)] = \alpha$  where  $\alpha$  is same for all users, then the resulting random access algorithms are called ALOHA algorithms
2. **Class  $\mathcal{A}_1^N$**  If the function  $f[i, j, \Theta(t), g_i^j(t)]$  takes only two values 0 or 1, then that class of algorithms is denoted by  $\mathcal{A}_1^N$ . Clearly, here we have  $g_i^j(t + 1) = f[i, j, \Theta(t), g_i^j(t)]$ . Thus the random access algorithm within this class can be represented (with a little abuse of notation) by the function  $f[i, j, \Theta(t)]$

**Remark 2.5.1** The random access algorithm within the class  $\mathcal{A}_1^N$  can be completely described by the sequence of sets  $\{B(t), t \in I\}$  where the set  $B(t)$  contains the addresses of users allowed to transmit the first packets of their buffer (if any) at the time instant  $t$ . For example at the instant  $t$ , if some  $m$  users are allowed to transmit (i.e., they are enabled), then set  $B(t) = \{i_1, i_2, \dots, i_m\}$ , where  $i_k \in \{1, 2, \dots, N\}$  and for those  $i_k \in B(t)$ ,  $f[i_k, j, \Theta(t), g_k^j(t)] = 1$  and  $j$  is the first packet in the buffer of user  $i_k$ .

**Remark 2.5.2** In the infinite user model, each packet can be considered as a user and the packet's generation instant can be mapped into the corresponding user's address. It is then possible to define a random access algorithm for the infinite user model using the same function  $f[i, j, \Theta(t), g_i^j(t)]$  where now  $i \in \{1, 2, \dots\}$  and  $j = 0$  or  $1$ . However, we have used the definition of Section 2.2.1 so as to conform to the literature.

## 2.5.2 Parameters for Characterization of Random Access Algorithm

Similar to that of an infinite user model, the parameters used to characterize a random access algorithm for a finite user model are delay, throughput and stability. The definitions of delay, throughput and stability are similar to those defined in Section 2.2.

- The packet delay for a finite user model, however, may be dependent upon the user's index. Thus let  $\delta_x(i)$  be the delay of a packet arriving at time  $x$  at the  $i$ th user, then the average packet delay at user indexed as  $i$  is given by

$$\mathcal{D}_1(i) = \lim_{x \rightarrow \infty} E[\delta_x(i)] \quad (2.34)$$

With the packets indexed as  $1, 2, \dots$  according to the order of their arrivals at the user  $i$ , let  $\delta_j(i)$  denote the delay experienced by the  $j$ th packet. Then the average packet delay at the user  $i$  is also defined by

$$\mathcal{D}_2(i) = \lim_{m \rightarrow \infty} \left[ \frac{1}{m} \sum_{j=1}^m \delta_j(i) \right] \quad (2.35)$$

- If we assume that packets arrive to each user's buffer according to a Poisson process of intensity  $\lambda/N$  such that the combined intensity is  $\lambda$ , then as in Section 2.2.3 the throughput can be defined as

$$\mathcal{T}_1 = \sup\{\lambda \mid \mathcal{D}(i) < \infty, \forall i\} \quad (2.36)$$

where  $\mathcal{D}(i)$  is any of the delays defined earlier at the user  $i$

The other definition of throughput given in Equation 2.7 can also be used here

- A finite user random access system is stable if  $\mathcal{T} > 0$ , otherwise it is unstable

The stability of a finite user model can also be defined in the following way. Let  $Q(t) = [Q_1(t), Q_2(t), \dots, Q_N(t)]$ , where  $Q_i(t)$  = Number of packets in the queue of  $i$ th user at time  $t$ . The system is said to be stable if the size of the queue remains bounded at each user as the system is allowed to operate for a long time. In other words a finite user random access system is stable if  $\lim_{m \rightarrow \infty} \lim_{t \rightarrow \infty} \text{Prob}\{Q_i(t) < m\} = 1 \forall i$ .

As in Section 2.3, the capacity of the system is defined as the supremum of throughputs achievable over all possible algorithms. The following proposition can then be proved [43] for a finite user random access system

**Proposition 2.1** *The capacity of a finite user model is one*

This fact can be illustrated as follows

Let us assume that packets arrive to each user according to a Poisson process of intensity  $\lambda/N$  such that the combined intensity is  $\lambda$ , then from [74] we note that the system is stable if the following condition is satisfied

$$\lambda < \left(1 - \frac{1}{N}\right)^{N-1}$$

Suppose TDMA strategy is used in a finite user model. In this strategy, there is a frame of  $N$  slots. At the beginning of each frame, each user is allotted one slot of the frame. The user transmits its first packet of the buffer in the

slot allocated to it. Thus a user is allowed to transmit only one packet per frame. Then it is obvious that the packets' queues would remain bounded with probability one if the total arrival intensity  $\lambda$  is kept less than one. Thus a TDMA strategy has a maximum throughput of one, *ie*, it achieves the capacity of a finite user model.

**Remark 2.5.3** Though TDMA seems to perform better than ALOHA in the sense that TDMA can accept an arrival rate up to 1 packet/slot, but the advantage of ALOHA is that when the packet arrival intensity is smaller than  $(1 - 1/N)^{N-1}$ , then ALOHA yields much lower average delay than that of TDMA. If the arrival rate is between  $\lambda = (1 - 1/N)^{N-1}$  and  $\lambda = 1$ , then one would prefer TDMA in comparison to ALOHA. The question that needs to be asked here is –*do better random access algorithms exist which can yield lower packet delay than that of TDMA for the range of arrival rates mentioned above?*

In a finite user random access model, it is therefore of interest to investigate stable random access algorithms which have better average delay characteristics.

### 2.5.3 Review of Random Access Algorithms for Finite User Model

We now provide below a brief review of two random access algorithms. These are

- Tree Algorithm of Capetanakis
- Group Testing Algorithms

These algorithms consist of a channel access and a collision resolution algorithm. The operation of the system can be divided into several collision resolution epochs. In addition it is assumed in these algorithms that the users have only two buffers. The first buffer contains the packet which undergoes the collision resolution in the current epoch while the second buffer contains any packet that might arrive during the epoch.

### Tree Algorithm

The tree algorithm of Capetanakis for a finite user model puts a restriction on the number of users in the system. The number of users in the system can only be a power of two. Let the number of users be  $N = 2^K$ . The users are addressed as  $1, 2, \dots, N$ . The users are assigned to the leaves of a binary tree. The algorithm assumes that each user can generate only one packet per epoch.

The depth of a node is defined [6] as the number of branches between the node and the root node. The depth of the root node is zero. The number of branches from a node is called the degree of a node. Let the nodes of the tree be denoted by  $n_{i,j}$ , where  $i \leq j \leq 2^i$  and  $0 \leq i \leq K$ . The algorithm operates as follows.

Initially all users are enabled or allowed to transmit. If there is no collision, then all users are resolved, otherwise, the users on top half of the tree retransmit in the very next slot and those on bottom half wait till any collision among the users on the top half have been resolved. The procedure is continued till there is no more collision to be resolved or the  $K$ th level of the tree is reached. The number of slots used in a collision resolution epoch is twice the number of nodes visited.

If ternary feedback is available, then Massey's modification [44] can be incorporated. In this modified tree algorithm if a collision slot due to the transmission of users is followed by an empty slot, then users on bottom half of the subtree under consideration are further divided into two parts.

Let  $p$  denote the probability that a user has a packet at the beginning of the collision resolution epoch. Let  $L(N; p)$  denote the expected length of the collision resolution epoch. Then as shown in [6], we have

$$L(N, p) = 1 + \sum_{i=1}^{N-1} 2^i \left\{ 1 - (1-p)^{2^{K-i}} - 2^{K-i} p (1-p)^{2^{K-i}-1} \right\} \quad (2.37)$$

In this equation  $N = 2^K$ . We note from this expression that the tree algorithm performs worse than TDMA if  $p > 1/\sqrt{2}$ . If the probability that a user has a packet is known, then Capetanakis has suggested an optimum tree protocol

This optimum tree protocol strives to minimise the expected number of slots needed to resolve collisions among the  $N = 2^k$  users. In the basic tree algorithm, the root node has degree  $2^k$ . Let  $2^{k'}$  be the degree of the node in an optimum tree protocol where  $k' = 1, 2, \dots, K$ . Capetanakis [6] has derived an expression for choosing the optimum value of  $k'$  which minimizes the expected length of the collision resolution epoch.

It can be shown [6] that the maximum average throughput for a binary tree algorithm with  $N = 2^k$  users and with the degree of the root node being equal to  $2^{k'}$  is given by

$$T = \frac{1}{2 - 2^{(k'-k)}} \quad (2.38)$$

### Group Testing Based Random Access Algorithms

Another random access algorithm that was applied to the finite user model was based upon the idea of group testing algorithms considered in statistics. The classical group testing methods are based upon classifying a set of objects into defective and nondefective. Let the set consist of  $M$  objects where  $p$  is the probability that an object from the set is defective and  $q = 1 - p$  is the probability that the object is good. Group testing consists of testing on a subset of this set. A *positive* result of this test means that all objects of this subset are nondefective while *negative* result means that at least one of the objects is defective.

One is interested in finding the optimal test strategy that minimizes the expected number of tests in classifying all objects. No optimal strategy, however, is known. Sobel and Groll [69] considered a restricted set of strategies known as *nested* strategies. In a nested strategy, if the result of the test is positive at some stage, then the next test is done on a proper subset of the set which was tested at the previous stage. The group testing algorithm described in this form lends naturally to its application in binary feedback of *something/nothing* kind. For a ternary feedback model, the algorithm was slightly modified and applied for a finite user model with Bernoulli arrival process. The steps of the algorithm are described in [85]. Like Capetanakis's

tree algorithm, this algorithm also performs worse than TDMA for  $p > 1/\sqrt{2}$ . The group testing algorithm, however, gives only a slight improvement over Capetanakis's optimum tree algorithm.

## 2.6 Random Access Systems with Multipacket Reception Capability

In the previous sections, we have considered random access algorithms for the conventional random access system which can be modelled by the common receiver model discussed in Section 2.1. It was assumed there that simultaneous transmission of two or more packets results in collision and destroys all the transmitted packets. In practical situations, however, because of the capture phenomenon, if the power of one of the received packets is sufficiently stronger than that of other packets, then the receiver may actually be able to correctly decode this packet. Code division multiple access (CDMA) used in random access packet radio network is another example. In CDMA-Random access system, the users encode their packets with pseudo-orthogonal codes. When the encoded packets are transmitted simultaneously in a slot, then due to the pseudo-orthogonality of the codes, some packets may be received successfully. The collided packets require retransmission and the users may employ some random access algorithm for rescheduling these packets. Such systems are called random access systems with multipacket reception capability.

Conceptually at the macro level, such system can be considered to be a generalization of conventional random access system and can be represented by the conditional probability matrix  $[Q_{n,k}]$  where  $Q_{n,k}$  is the probability that  $k$  packets are received successfully given  $n$  packets are transmitted in a slot. Such a conceptual  $(n,k)$  channel was considered by Sylvie Ghez *et al* [24]. In [47], [30], [61] a  $d$ -channel model of this system has been considered. In this model, if more than  $d$ -packets are transmitted in a slot, all are destroyed. But if less than  $d$ -packets are transmitted, then all packets can be received successfully. This  $d$ -channel model is in fact a special case of  $(n,k)$  channel.

model

We have seen that though in an actual practical system, there will be many users, each with a transmitter and a receiver, trying to send messages, this system can be modelled by the common receiver model which has been discussed in detail in Section 2.1. In a generalized random access channel, however, there are various other factors which come into picture and the system model has to be considered at the *macro* level as opposed to the *micro* level of conventional random access channel. What follows therefore is the general description of this system model. Our discussion below is with respect to CDMA-RA system.

### 2.6.1 CDMA-RA System Model

In CDMA-RA system discussed here, we assume slotted operation. Note that in conventional random access system, the slotted operation may increase the throughput of some random access algorithms. For example, slotted ALOHA has double the throughput of unslotted ALOHA. However, when CDMA is used in a network then because of the graceful degradation characteristics [82] of CDMA system, the difference in performance between slotted and unslotted system is not very significant [58]. In such systems, to achieve reception of multiple packets, the packets are transmitted by encoding them with codes. These codes are chosen from sets of codes with low cross-correlation properties. The codes used to encode packets are called address codes. Various schemes are possible for assigning these addresses. These are.

- 1 *Transmitter Based Coding* In transmitter based coding, each user is assigned a fixed address code for transmission. The user encodes its packet with this address code before transmitting it. The different address codes are pseudo-orthogonal to each other. Each receiver has the decoder for all address codes. Instead of fixed address coding, random address coding may also be used. In random address coding, each user's transmitter is provided with a set of orthogonal codes. The number of such codes is much smaller than the number of users in the system. Each user before transmitting its packet, randomly chooses one of the codes and then uses it as its address. If two or more users happen to choose the same code,

then their packets collide

- 2 *Receiver Based Coding* In receiver based coding, a unique receiving code is assigned to each user. Any user which wants to send its packet to a particular receiver encodes its packet with the address code of that receiver and then transmits. If two or more users send packets to the same receiver simultaneously, the packets collide and are destroyed. This collision is called primary level interference. Secondary level interference may also occur due to the pseudo-orthogonality of addresses of different receivers.

Combination of the above schemes are also possible. In transmitter based code assignment with fixed address codes, the source of a packet is usually identified by the address of the user. In the case of random address codes used in transmitter based assignment and also in receiver based assignment, a packet may contain a header identifying the source of the packet. The packets may also use some form of error correcting codes.

There are various types of address codes that can be used in CDMA-RA system. One such scheme is pulse position tree coded multiple access (TCMA). The random access system using TCMA as the transmitter based coding was analyzed by Raychaudhuri [60]. In TCMA, each bit of the packet is encoded using a convolutional tree code of constraint length, say  $K$ . The output symbols are one of the  $2^K$  orthogonal pulses for every bit. The bandwidth expansion factor then is  $2^K$ . The details of this scheme are given in [60]. Goodman *et al* [26] and Einarsson [13] have proposed coding schemes that use some form of time-frequency coding. These schemes can also be used as the address coding in CDMA-RA system.

The feedback information to the users about the collided packets is another important issue. In transmitter based coding, each user while transmitting the packet, may monitor the channel with this address. Any possible collision can thus be detected. In a system with common receiver, the receiver having decoders for all codes broadcasts the feedback information on a separate feedback channel. This feedback informs users about the status on each code. It is thus a form of *multibit feedback*. In receiver based coding, the destination

receiver may send acknowledgement signals to the transmitter which had sent the packet. We now briefly examine below some of the schemes discussed in the literature.

### 2.6.2 Review of Random Access Algorithms for Multipacket Reception Channels

If we assume that there are many users trying to send packets to a single receiver and these users employ transmitter based address coding with the receiver broadcasting the feedback on a separate feedback channel, then this model is a straightforward generalization of the common receiver model of Section 2.1. Accordingly in this section, we restrict our attention to this generalized common receiver model and review some of the random access algorithms considered for this model in the literature.

One of the first studies in CDMA-RA system was done by Raychaudhuri [60]. Two types of multiaccess coding techniques, namely Time Hopping Multiple Access and Tree Coded Multiple Access, are considered for address codes. These addresses are uniquely assigned to each transmitter. The random access algorithm for the retransmission of collided packets is ALOHA algorithm. Steady state throughput analysis was performed for this scheme. The method of analysis is similar to that of ALOHA for conventional random access channel. One of the important conclusions of this study is that with TCMA as the address coding, it is possible to achieve higher normalized throughput than that of ALOHA for normal channel. The system is found to exhibit similar kind of stability behavior. More importantly, the performance degradation of this CDMA-RA system for infinite user model with TCMA as the address coding is found to be more graceful in the sense that the delay does not increase as rapidly as in the normal ALOHA when the system is operated beyond the maximum stable throughput. The system, however, exhibits the similar kind of bistable channel behavior as is observed in ALOHA for a finite user model. This system therefore needs appropriate control schemes. The control schemes for such systems may not only depend upon choosing the proper retransmission probability but there are also other parameters like code

rate *etc* that can be varied. These additional degree of freedom available add new dimension to the control problem.

Pursley [59] studied the performance of frequency hop transmission in random access system. Sylvie Ghez *et al* [24] studied a generalized random access system. This model is characterized by the probability matrix  $[Q_{n,k}]$  where  $Q_{n,k}$  is the probability that  $k$  packets are received successfully when  $n$  packets are transmitted. Let  $C_n = \sum_{k=1}^n kQ_{n,k}$  be the average number of packets successfully received when the number of packets transmitted is  $n$ . Let the average arrival rate of packets be  $\lambda$ . Let  $\zeta_0 = \lim_{n \rightarrow \infty} C_n$  (assuming this limit to exist). Suppose ALOHA is used as the random access algorithm for the retransmission of collided packets. Then it was proved in [24] that the system is stable for all arrival rate distributions such that  $\lambda < \zeta_0$  and is unstable for  $\lambda > \zeta_0$ . If  $\zeta_0$  is infinite, then the system is always stable. The control scheme based on varying the retransmission probability was also considered by the authors in a subsequent paper [25].

Random access algorithms employing collision resolution strategies were considered in CDMA environment by Hu and Chang [30] and Mehravari [47]. Hu and Chang's work [30] can be considered as an extension of the tree algorithm to transmitter based address assignment scheme.

## Chapter 3

# An Optimal Collision Resolution Algorithm for Single Buffer Users

---

In this chapter, we introduce the random access model which will be the model for our investigation. The basic assumptions used in this model are similar to those of the finite user model of Chapter 2. For this finite user random access model, we focus our attention on the blocked random access algorithms. As defined in Chapter 2, in a blocked random access algorithm, the users are prevented from transmitting any new packet till the contending users have resolved their collisions using a collision resolution algorithm. The time interval during which the contending users resolve their collisions is called the collision resolution epoch. All communicating users are considered identical here and the users as decision makers follow identical decision rules to resolve collisions, if any. These decisions are to be made in a decentralized way and the users share the common objective of utilizing the channel as efficiently as possible. This leads to formulating the problem of random access algorithm within the framework of decentralized team decision theory.

For a finite user model, we address ourselves the question of designing an optimal collision resolution algorithm in a blocked random access environment. To facilitate our investigations, we assume that initial collision multiplicity or the number of active users at the beginning of a collision resolution epoch is known. This assumption, though somewhat restrictive, has been made here with a view to gain an insight into the problem. It is relaxed in Chapter

4, where we consider a more general model. However, as will be shown in this chapter, the assumption about knowing the number of active users is not a completely unreasonable one in practice. Indeed this assumption has also been made by other authors [21, 55] in the context of different collision resolution algorithms.

Within the above framework, we identify an appropriate set of states and control variables that constitute a Markovian Decision Process (MDP) for the random access system. It is then argued that the problem of optimal collision resolution algorithm is equivalent to the first passage problem of this MDP. The dynamic programming technique is applied to derive an algorithm which is optimal in the sense of minimizing the expected time of termination or collision resolution epoch length when we know the number of active users at the beginning. Finally, we consider the steady state operation of our system under this protocol and suggest an approximate method of estimating the steady state expected collision resolution length.

### 3.1 The System Model

We consider the finite user common receiver model as shown in Figure 2.2. The following assumptions are made about this model. Many of these assumptions are similar to those of Chapter 2.

1. There are  $N$  users in the system. These users are addressed as  $1, 2, \dots, N$ .
2. *Slotted Operation*: The channel is assumed to be time slotted. The users can start the transmission of their packets only at the beginning of a slot. The length of a slot equals the time required for the transmission of each fixed length packet. We assume here that each time slot is of one unit duration.
3. *Ternary Feedback*: If no packet is sent in a slot, then the channel is idle. On the other hand, if only one user transmits its packet, then the packet is successfully transmitted. However, when two or more users try to send their packets simultaneously in a slot, these packets then collide.

The users can distinguish between these three cases. In other words, it is assumed that the users receive ternary feedback 0, 1 or 2 depending upon whether the previous slot contained no packet, one packet or more than one packet respectively. The feedback is assumed to be immediately available at the end of the slot. It is also assumed that the feedback information is received errorless by all users.

- 4 *Blocked Access Operation* We assume that the users employ blocked random access algorithm. In this model, the operation of the system can be divided into successive transmission intervals. In each transmission interval, users resolve their collisions, if any, by using a collision resolution algorithm. This transmission interval is called the collision resolution epoch. The users are prohibited from transmitting any new packet that might have arrived during this epoch. The packets generated in one epoch are transmitted in the next epoch. Thus a collision resolution epoch ends, when all users have transmitted successfully their packets that were generated in the previous epoch.
- 5 *Single Packet per Epoch* We assume that each user can generate only one packet during a collision resolution epoch. An alternative way of assuming this may be that each user has only two buffers. The first buffer contains the packet which is undergoing resolution in the current epoch. The second buffer contains any new packet that might arrive at the user during the epoch.<sup>1</sup> All subsequent packets arriving after this packet which is stored in the second buffer, are lost. At the end of the current epoch, packets in the second buffer are moved into the first buffer and are transmitted.

Let  $\sigma$  be the probability with which a user generates a packet in each slot during a collision resolution epoch. If  $\ell$  is the length of the collision resolution epoch, then the probability that a user has a packet at the end of the epoch will be  $p = 1 - (1 - \sigma)^\ell$ . For sufficiently small values of  $\sigma$ , even with large collision resolution length  $\ell$ ,  $p$  can be assumed to be

---

<sup>1</sup>Since a user has only one buffer to store a new packet, it is usual to refer to this case as the single buffer.

the probability of generating one packet during the epoch. Such packet generation model has also been considered by Capetanakis [6].

Although we have considered users with single buffers, we can still apply this scheme to users with infinite buffer capacity but with the restriction that each user is allowed to transmit only one packet per epoch. A more efficient protocol in which users are allowed to transmit more than one packet per epoch can also be considered. The steady state analysis under dynamic operation of this protocol for buffered users is considered in Chapter 5.

## 6 Active Users

**Definition 3.1** *A user is defined to be an active user if it has a packet waiting in its buffer for transmission.*

We assume that the number of active users at the beginning of a collision resolution epoch is known to all the users. The users may acquire this information in the following ways:

- (a) The common receiver may broadcast the information about the number of active users on a separate feedback channel (this channel may be a low data rate channel).
- (b) Alternatively, all users may be enabled initially. The users will transmit their packets, if any. If two or more users are active, it will result in collision. The users are equipped with receivers having energy detectors. With the help of energy detectors, the users measure the initial collision multiplicity or the number of active users.

Such an assumption has also been made by other authors in the literature [21]. In either case, the users after knowing the number of active users, enter a collision resolution epoch.

### 3.2 Random Access Algorithm for Finite User Model

In the previous chapter, we have defined a random access algorithm for the finite user model. Let us consider the set  $B(t) = \{i \mid f[i, j, \Theta(t), g_i^j(t)] = 1\}$  where  $i$  is the user's address, i.e.,  $i \in \{1, 2, \dots, N\}$  and  $j$  is the index of the first packet of the user  $i$  at time instant  $t$ . In other words, the set  $B(t)$  is the set of users allowed to transmit the first packets of their buffers at the time instant  $t$ . Then as seen in Section 2.5.1, the random access algorithm for a finite user model within the class  $\mathcal{A}_1^N$  can be described by the sequence of sets  $\{B(t), t = 0, 1, \dots\}$ .

In this chapter, we want to consider those random access algorithms within the class  $\mathcal{A}_1^N$  which employ the blocked channel access. Thus we now need to consider only collision resolution algorithm to completely characterize our random access algorithm. Accordingly, we limit our attention to the operation of collision resolution algorithm during a single collision resolution epoch. Further, as stated earlier, we assume that a user is allowed to transmit only one packet per epoch.

Let us now consider a particular collision resolution epoch. The length of each slot is of one time unit as assumed earlier in Section 3.1. Then the collision resolution algorithm within that epoch can be specified by the set  $\{B(t)\}$  where  $B(t) = \{i \mid f[i, \Theta(t)] = 1\}$ ,  $t \in I = \{0, 1, \dots\}$  and  $t = 0$  denote the beginning of the collision resolution epoch under consideration. Our interest in this chapter is in the design of function  $f$  which characterizes the collision resolution algorithm. This function in effect specifies the set of users allowed to transmit their packets at each stage. For this, we first study the dynamics of the system in the following section.

### 3.3 Dynamics of the System

#### 3.3.1 The System State

**Definition 3.2** *A user is said to have been resolved if one of the following conditions is obtained*

- 1 *All other users know that the user does not have a packet*
- 2 *The user was enabled in a slot that was either empty or successful*

Note that the second condition implies that although the identity of the user having a packet does not become known to other users, all those users which were enabled in a slot that resulted in a success are treated as resolved. A user which is not resolved, is treated as unresolved. We now define the state of our system as follows

**Definition 3.3** *The state of the system at the instant  $t$  is defined to be a two tuple  $S(t) = [N(t), n(t)]$ , where*

$N(t)$  = Number of unresolved users at time  $t$

$n(t)$  = Number of active users at time  $t$

Since at the beginning of the collision resolution epoch, all users in the system are unresolved, we have  $N(0) = N$ . If the number of active users at the beginning of the epoch is some number say  $n$ , i.e.,  $n(0) = n$ , then the initial state is  $S(0) = (N, n)$ . Note that  $(0, 0)$  is the state at which all users are resolved. By definition therefore, the collision resolution epoch ends when the state  $(0, 0)$  is reached. In our model, the initial state  $S(0) = (N, n)$  is assumed to be known to all users.

#### 3.3.2 Enumeration of the States

Let  $S$  denote the set of all possible states. Let the number of possible states be denoted by  $\mathcal{L}$ . We then have the following proposition

**Proposition 3.1** *If  $N$  is the total number of users in the system, then the total number of possible states will be  $\mathcal{L} = N(N + 1)/2 + 1$*

*Proof* The proof is straightforward and omitted here

Let  $S_i = (N_i, n_i)$  denote the  $i$ th state. Here  $N_i$  may be any number from 0 to  $N$  and  $n_i \leq N_i$ . Thus the state space  $\mathcal{S} = \{S_1, S_2, \dots, S_{\mathcal{L}}\}$ , where  $S_1 = (N_1, n_1)$ ,  $S_2 = (N_2, n_2)$  and so on. If the number of users in the system is  $N = 8$ , then the possible states are listed in Table 3.1 (state (0,0) is not listed)

The evolution of this system state with respect to a collision resolution algorithm is discussed in the next subsection

### 3.3.3 Evolution of the System State

A collision resolution algorithm can be defined as a control policy which comprises a sequence of actions to be taken at each stage and each such action, specifies the set of users to be enabled for transmission at that stage. This control policy may depend upon the state and the history of the system. If the number of unresolved users is  $N_i$ , then the policy can be described as specifying at each stage an identical  $N_i$  dimensional vector which is used by all users at that stage. Each element of this vector assumes binary values of either 0 or 1. If the  $i$ th element of this vector at any stage is 1, then the corresponding user is allowed to transmit at that stage.

The control policy considered in this chapter gives the optimum number of users to be enabled at each stage. Let  $\{S(t), t = 0, 1, \dots\}$  denote the sequence of observed states. Similarly let  $\{u(t), t = 0, 1, \dots\}$  denote the sequence of control actions taken. Thus the control  $u(t)$  here specifies the number of users to be enabled during slot  $(t, t + 1)$ . Let  $\mathcal{U}(S(t))$  denote the set of admissible control actions at time  $t$ . We will also use the notation  $\mathcal{U}(S_i)$  to denote the set of admissible control actions when the state is  $S_i = (N_i, n_i)$ . Since  $N_i$  is the total number of users to be resolved, the set of admissible control actions when the state is  $S_i$  will be  $\{0, 1, \dots, N_i\}$ .

If the number of unresolved users is  $N_i$ , then there will be exactly  $\binom{N_i}{u}$  possible control vectors for control  $u$ . The users may agree upon a predetermined

protocol to decide upon the particular vector so that each user chooses the same vector at its site. The users may use a pseudorandom number generator with the same seed to choose the vector.

Assume that at the instant  $t$ ,  $S(t) = S_i = (N_i, n_i)$  where  $n_i > 2$  and control  $u(t) = u$ . If now  $u$  users are enabled and an empty feedback is observed then it is known that the enabled  $u$  users do not have packets and they are said to be resolved, so that the next state  $S(t+1)$  moves to  $(N_i - u, n_i)$ . On the other hand, if  $u$  users are enabled to transmit and it results in success, then the next state is  $(N_i - u, n_i - 1)$ . However, if this action results in collision, then no user is resolved and the next state  $S(t+1)$  continues to have the same value as that of  $S(t)$ .

On the other hand, assume now that  $n_i = 2$  and all users have perfect knowledge of the state  $(N_i, n_i)$ . If now collision takes place by enabling  $u$  users, then the users would come to know that the remaining  $(N_i - u)$  users do not have packets, so they can be treated as resolved and the next state is  $(u, 2)$ . If however, an empty feedback is observed, then the next state is  $(N_i - u, 2)$  and if a success is observed, then the next state is  $(N_i - u, 1)$ .

**Remark 3.3.1** We note that if the state  $S(t)$  is known perfectly, then by observing the feedback  $\theta(t+1)$  and the control action  $u(t)$ , the users can have perfect knowledge of the state  $S(t+1)$ . Since the initial state  $S(0)$  is assumed to be known to all users, it follows that users have perfect knowledge of the state at every stage.

**Remark 3.3.2** We also note that the probability of transition from the present state  $S(t) = S_i = (N_i, n_i)$  to the next state  $S(t+1) = S_j = (N_j, n_j)$  depends only upon the state  $S(t) = S_i = (N_i, n_i)$  and the control action  $u(t) = u$ . The following proposition can thus be stated.

**Proposition 3.2** For a given sequence of control actions,  $u(0), u(1), \dots, u(t)$ , the sequence  $\{S(t), t \in I\}$  is a Markov chain, i.e.,

$$\begin{aligned} \text{Prob}[S(t+1) = S_j \mid S(t) = S_i, u(t) = u, S(t-1), u(t-1), \dots, S(0), u(0)] = \\ \text{Prob}[S(t+1) = S_j \mid S(t) = S_i, u(t) = u] \quad (3.1) \end{aligned}$$

Let us denote this one step state transition probability by  $p_{S_i, S_j}(u)$ . Let  $P(u) = \left[ p_{S_i, S_j}(u) \right]$  denote the state transition probability matrix with elements  $p_{S_i, S_j}(u)$  for  $S_i, S_j \in \mathcal{S}$ .

As defined in [11, 5, 64] Markovian Decision Process is a class of Stochastic Sequential Decision Processes where the transition probability function depends only upon the present state and the present control action. Thus the stochastic process  $\{S(t), u(t), t \in I\}$  is a Markovian Decision Process. The control  $u(t)$  may depend upon the entire history of the system. If the control  $u(t)$  depends only upon the state  $S(t)$ , then clearly the sequence  $\{S(t), u(t), t \in I\}$  is a Markov chain. Finally, we can conclude our discussion of this subsection in the following proposition.

**Proposition 3.3** *The finite user random access model of Figure 2.2 is completely characterized by 4-tuple  $\{\mathcal{S}, \mathcal{U}, S(0) = (N, n), \{P(u), \forall u \in \mathcal{U}\}\}$  together with the property 3.1, where  $\mathcal{S}$  is the finite state space,  $\mathcal{U}$  is the collection of the sets of admissible control actions for all states, i.e.,  $\mathcal{U} = \{\mathcal{U}(S_i) \mid \forall S_i \in \mathcal{S}\}$ ,  $P(u)$  is the state transition probability matrix when the control applied is  $u$ ,  $\{P(u)\}$  is the collection of such probability matrices and  $S(0) = (N, n)$  is the initial state assumed to be known to all the users at the beginning of the collision resolution epoch.*

### 3.3.4 Evaluation of State Transition Probabilities

In this section, we calculate the probability  $p_{S_i, S_j}(u)$  of transition from the state  $S_i = (N_i, n_i)$  to another state  $S_j = (N_j, n_j)$  when control action  $u$  is employed.

Let the state at any instant be  $S_i = (N_i, n_i)$  where  $n_i > 2$ . If control  $u$  is applied, then the next state can be either  $(N_i, n_i)$  or  $(N_i - u, n_i)$  or  $(N_i - u, n_i - 1)$ . The transition from  $(N_i, n_i)$  to  $(N_i - u, n_i)$  occurs when by enabling  $u$  users, chosen randomly from amongst  $N_i$  users, an empty feedback is observed. Similarly, the transition from  $(N_i, n_i)$  to  $(N_i - u, n_i - 1)$  occurs, when success is observed, i.e., among the enabled  $u$  users, only one user had a packet. The

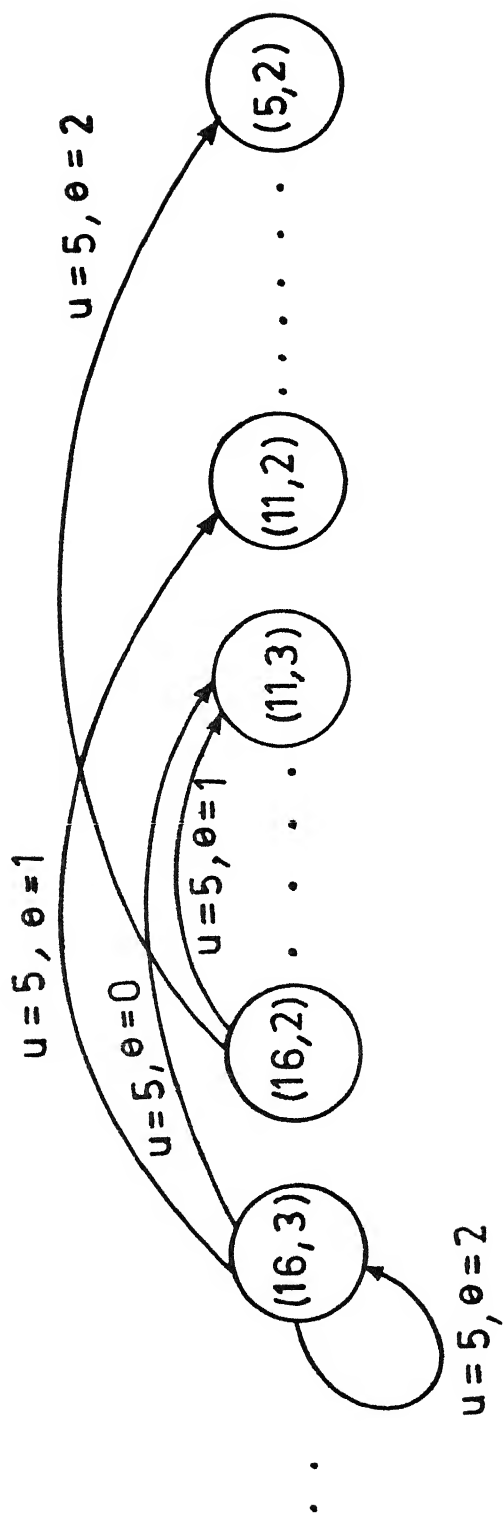


FIG. Evolution of the states with control actions for a representative case

probabilities of state transition for  $n_i > 2$  are given by the following relations

$$p_{(N_i, n_i), (N_i - u, n_i)}(u) = \frac{\binom{u}{0} \binom{N_i - u}{n_i}}{\binom{N_i}{n_i}} \quad (3.2)$$

$$p_{(N_i, n_i), (N_i - u, n_i - 1)}(u) = \frac{\binom{u}{1} \binom{N_i - u}{n_i - 1}}{\binom{N_i}{n_i}} \quad (3.3)$$

However, if among the enabled  $u$  users, two or more users have packets then none of the users are resolved. We thus have

$$p_{(N_i, n_i), (N_i, n_i)}(u) = \frac{\sum_{r=2}^u \binom{u}{r} \binom{N_i - u}{n_i - r}}{\binom{N_i}{n_i}} \quad (3.4)$$

The probability of transition from  $(N_i, n_i)$  to all other states will be zero. This fact can be expressed as

$$p_{(N_i, n_i), (N_j, n_j)}(u) = 0 \quad \text{for } N_j \neq N_i \text{ or } (N_i - u) \text{ and } n_j \neq n_i \text{ or } (n_i - 1) \quad (3.5)$$

If  $n_i = 2$ , then the next state can be  $(N_i - u, 2)$  or  $(N_i - u, 1)$  or  $(u, 2)$ . The state transition probabilities from the state  $(N_i, 2)$  to  $(N_i - u, 2)$ ,  $(N_i - u, 1)$  and  $(u, 2)$  are given as follows

$$p_{(N_i, 2), (N_i - u, 2)}(u) = \frac{\binom{u}{0} \binom{N_i - u}{2}}{\binom{N_i}{2}} \quad (3.6)$$

$$p_{(N_i, 2), (N_i - u, 1)}(u) = \frac{\binom{u}{1} \binom{N_i - u}{1}}{\binom{N_i}{2}} \quad (3.7)$$

$$p_{(N_i, 2), (u, 2)}(u) = \frac{\binom{u}{2} \binom{N_i - u}{0}}{\binom{N_i}{2}} \quad (3.8)$$

In the above equations, it was assumed that  $N_i - u \neq u$ . However, if  $N_i - u = u$ , then the next state can only be  $(u, 2)$  or  $(N_i - u, 1)$ . The probability of transition from  $(N_i, 2)$  to  $(N_i - u, 1)$  will be the same as above but the transition probability from the state  $(N_i, 2)$  to  $(u, 2)$  will be given by

$$p_{(N_i, 2), (u, 2)}(u) = 2 \frac{\binom{u}{2} \binom{N_i - u}{0}}{\binom{N_i}{2}} \quad (3.9)$$

Let  $\Phi$  be an admissible control policy which specifies a control action corresponding to each state. The policy  $\Phi$  may specify some control (say  $u \in \mathcal{U}(S_i)$ ) to be applied when the state is  $S_i$ . We then have the following theorem.

**Theorem 3.1** Let  $p_{(N_i, n_i), (0,0)}^{(k)}(\Phi)$  denote the  $k$ -step state transition probability from  $(N_i, n_i)$  to  $(0,0)$  when controls at various stages are specified by the policy  $\Phi$ , i.e.,

$$p_{(N_i, n_i), (0,0)}^{(k)}(\Phi) = \text{Prob}[S(k) = (0,0) \mid S(0) = (N_i, n_i), \Phi] \quad (3.10)$$

then  $p_{(N_i, n_i), (0,0)}^{(k)}(\Phi) > 0$  for some  $k \geq 1 \forall S_i = (N_i, n_i) \in \mathcal{S}$

*Proof* The  $k$ -step transition probability  $p_{(N_i, n_i), (0,0)}^{(k)}(\Phi)$  can be written as

$$p_{(N_i, n_i), (0,0)}^{(k)}(\Phi) = \sum_{S_j = (N_j, n_j) \in \mathcal{S}} p_{(N_i, n_i), (N_j, n_j)}(\Phi) p_{(N_j, n_j), (0,0)}^{(k-1)}(\Phi) \quad (3.11)$$

Assume that an admissible control  $u(0) = u$  is applied at the time  $t = 0$ . Since the state  $S(0) = (N_i, n_i)$ , the set of admissible controls will be  $\{0, 1, \dots, N_i\}$ . Then the next state moves to some other state  $S_j = (N_j, n_j)$ . From the expressions for the state transition probabilities, we know that  $S_j$  has the nonzero probability of being one of the following states  $(N_i, n_i), (N_i - 1, n_i), \dots, (n_i, n_i), (N_i - 1, n_i - 1), (N_i - 2, n_i - 1), \dots, (n_i - 1, n_i - 1)$  depending upon the control specified by the policy. If  $S_j = (0,0)$  then  $p_{(N_i, n_i), (0,0)}^{(k)}(\Phi) = 1$  for  $k = 1$ . However, if  $S_j \neq (0,0)$  then the control corresponding to the state  $S_j$  is applied at the stage. It follows easily that ultimately we can reach the state  $(0,0)$  in some finite number of steps  $k$  and the  $k$ -step transition probability can be calculated in a recursive manner by Equation 3.11  $\square$

**Remark 3.3.3** For a given control policy, we may represent the state transitions at various stages by a graph in which nodes may correspond to the states and a directed arc from one node to another node represents the transition from one state to another state, when control action corresponding to the state as specified by the policy is applied. If  $p_{S_i, S_j}(u) = 0$ , then there will be no path from the node  $S_i$  to the node  $S_j$ . Then what the theorem states is that there exists a path from every possible initial state to the state  $(0,0)$  for any admissible control policy.

### 3.4 The Optimal Collision Resolution Algorithm

We are now in a position to formulate the problem of optimal collision resolution algorithm. The length of a collision resolution epoch is called the collision resolution length. The collision resolution algorithm is a control policy and the optimality criterion used is the expected collision resolution length.

The policy derived here gives the optimum number of users to be enabled at each stage. Any such control policy or strategy can be defined to be a set of functions

$$\{\phi_t[u | \mathbf{H}(t-1), \mathbf{S}(t)], t = 0, 1, \dots\}$$

where  $\phi_t[u | \mathbf{H}(t-1), \mathbf{S}(t)]$  is the probability with which  $u$  users are enabled at time  $t$  given the past history  $\mathbf{H}(t-1)$  and the state of the system  $\mathbf{S}(t)$ . The past history may consist of the past states and control actions, i.e.,

$$\mathbf{H}(t-1) = \{\mathbf{S}(0), u(0), \mathbf{S}(1), u(1), \dots, \mathbf{S}(t-1), u(t-1)\} \quad (3.12)$$

Since given the initial state, the users have perfect knowledge of the state at subsequent stages by observing the feedback and the control, the history vector  $\mathbf{H}(t-1)$  can be written as

$$\mathbf{H}(t-1) = \{\mathbf{S}(0), u(0), \theta(1), u(1), \theta(2), \dots, u(t-1)\}$$

If our strategy is such that the function  $\phi$  is independent of time  $t$  and the history of the system  $\mathbf{H}(t-1)$ , but depends only upon the value of the present state  $\mathbf{S}(t)$ , then such strategy is called a deterministic and stationary strategy. For  $\phi_t[u(t) = u | \mathbf{H}(t-1), \mathbf{S}(t)]$  to be deterministic and stationary function we have

$$\phi_t[u(t) = u | \mathbf{H}(t-1), \mathbf{S}(t) = \mathbf{S}_i] = \phi[u(t) = u | \mathbf{S}(t) = \mathbf{S}_i]$$

Thus  $\phi: \mathcal{S} \mapsto \mathcal{U}$  is a single valued map from the state space to the control space. Our deterministic and stationary policy  $\Phi$  will be given by

$$\Phi = \{\phi[\cdot], \phi[\cdot], \dots\}$$

Let  $\mathcal{Y}_D = \{\Phi\}$  be the class of all such deterministic stationary policies. We will see in Section 3.4.2 that our optimal policy belongs to this class. This

optimal policy specifies the number of users to be enabled for each state. As already explained, the precise set of users is to be chosen randomly. After the addresses of the users to be enabled have been determined, the function  $f[\cdot]$  characterizing our collision resolution algorithm is completely specified.

### 3.4.1 Statement of the Problem

We first introduce a cost structure into our model. Suppose a policy  $\Phi = \{\phi_0[\cdot], \phi_1[\cdot], \dots\}$  is used. After observing the state  $S(t)$ , which may be equal to  $S_i = (N_i, n_i)$ , an action  $u$  (specified by the policy) is taken. This state then undergoes a transition to the state  $S(t+1) = S_j = (N_j, n_j)$ . We assume that an incremental cost  $\psi(S(t+1) = S_j, S(t) = S_i, u(t) = u)$  is incurred with this transition. Define the following expected cost

$$\Psi(S_i, u) = \sum_{S_j \in \mathcal{S}} \psi(S(t+1) = S_j, S(t) = S_i, u(t) = u) p_{S_i, S_j}(u(t) = u) \quad (3.13)$$

Let  $\Psi(S(t), u(t))$  denote the cost incurred at time  $t$ . Then  $\Psi(S(t), u(t)) = \Psi(S_i, u)$  if at the instant  $t$ , the control  $u(t) = u$  and the state  $S(t) = S_i = (N_i, n_i)$ .

The total expected cost of operating the system up to the time  $t = \tau$  given that the initial state is  $S(0) = (N, n)$  and the policy used is  $\Phi$  is then given by

$$\begin{aligned} C_{\Phi, \tau}(S(0) = (N, n)) &= E \left[ \sum_{t=0}^{\tau} \Psi(S(t), u(t)) \right] \\ &= \sum_{t=0}^{\tau} \sum_{S_i \in \mathcal{S}} \sum_{u \in \mathcal{U}(S_i)} \text{Prob}[S(t) = S_i, u(t) = u \mid S(0) = (N, n)] \\ &\quad \Psi(S_i = (N_i, n_i), u) \end{aligned} \quad (3.14)$$

The problem of optimal collision resolution algorithm is as follows

Define the cost  $\Psi$  as follows

$$\begin{aligned} \Psi(S_i, u) &= 1 \quad \forall S_i \in \mathcal{S} \quad \text{except for the state } (0,0) \\ \text{and } \Psi((0,0), u) &= 0 \end{aligned} \quad (3.15)$$

Let  $\tau$  be the time instant when for the first time the state  $(0,0)$  is reached, i.e.,  $S(\tau) = (0,0)$  and  $S(t) \neq (0,0)$  for  $t < \tau$ . When the state  $(0,0)$  is reached, then

all users are resolved and the collision resolution epoch ends. Assume that  $p_{(0,0),(0,0)}^{(\Phi)} = 1$ , i.e., if the system reaches the state  $(0,0)$ , it remains there for any policy. We can then remove the dependence of the cost function  $C_{\Phi_\tau}$  on  $\tau$  and write the expression for it in the following form

$$\begin{aligned} C_{\Phi}(S(0) = (N, n)) &= E \left[ \sum_{t=0}^{\infty} \psi(t) \right] \\ &= \sum_{t=0}^{\infty} \sum_{S_i \in \mathcal{S}} \sum_{u \in \mathcal{U}(S_i)} \text{Prob}[S(t) = S_i, u(t) = u \mid S(0) = (N, n)] \\ &\quad \Psi(S_i = (N_i, n_i), u) \end{aligned} \quad (3.16)$$

Then the above cost  $C_{\Phi}(S(0) = (N, n))$  gives the expected collision resolution length. We are interested in collision resolution algorithm which minimizes the expected collision resolution length. In other words, our interest is in policy which minimizes the expected time to reach the target state  $(0,0)$ . We immediately see that the problem is equivalent to the optimal first passage problem of a Markovian Decision Process [11].

### 3.4.2 Existence of Optimal Collision Resolution Algorithm

As the problem of optimal collision resolution algorithm is equivalent to the first passage problem of a Markovian Decision Process, we can show that there exists an algorithm  $\Phi^*$  which is optimal. This optimal algorithm is a stationary deterministic policy.

Let  $\mathcal{Y}$  denote the class of all possible policies having dependence upon the entire history of the system. Let  $\mathcal{Y}_D$  denote the subclass of stationary deterministic policies (i.e.,  $\mathcal{Y}_D \subseteq \mathcal{Y}$ ). In a policy belonging to this subclass, the action at the time  $t$  depends only upon the state of the system at the time  $t$  and the policy defines a single valued transformation from the state space to the control space. We then have the following theorem.

**Theorem 3.2** *Let the initial state be  $S(0) = (N, n)$ . We define the cost structure as given in Equation 3.15. Since  $p_{(N,n),(0,0)}^{(k)}(\Phi) > 0$  for some  $k \geq 1$ , there exists an optimal policy  $\Phi^* \in \mathcal{Y}_D$  such that  $\Phi = \Phi^*$  minimizes the expected collision resolution length  $C_{\Phi}(S(0) = (N, n))$ .*

*Proof* This theorem is an obvious generalization of the existence theorem of the optimal first passage problem proved in [11] This proof is given in Appendix A  $\square$

Let this optimal expected collision resolution length be denoted by  $L^c(N, n)$ ,  
 $i.e., L^c(N, n) = \inf_{\Phi \in \mathcal{C}_D} C_{\Phi}(S(0) = (N, n))$

### 3.4.3 Determination of Optimal Collision Resolution Algorithm

In this section, we show how to determine an optimal collision resolution algorithm using dynamic programming techniques We have the following theorem which enables us to compute the optimal policy iteratively

**Theorem 3.3** Let  $V_0(S_i) = 0 \quad \forall S_i \in [\{S\} - (0, 0)]$  Define the following

$$V_{m+1}(S_i) = \min_{u \in \mathcal{U}(S_i)} \left[ \Psi(S_i, u) + \sum_{\substack{S_j \in \delta \\ S_j \neq (0,0)}} p_{S_i, S_j}(u) V_m(S_j) \right] \quad (3.17)$$

then  $\lim_{m \rightarrow \infty} V_m(S_i) = C_{\Phi^*}(S(0) = (N_i, n_i)) = L^c(N_i, n_i)$  where  $\Phi^* \in \mathcal{Y}_D$  minimizes  $C_{\Phi}(S(0) = (N, n))$

*Proof* This theorem can be proved as in [11] and is given in Appendix A  $\square$

Similar to Remark 3.3.3, the state transitions at various stages may be represented by a graph in which the nodes may correspond to the states The different outgoing arcs from a node may correspond to the controls admissible at the corresponding state The length of an arc from one node to another may correspond to the cost per stage The problem of finding an optimal collision resolution algorithm is then equivalent to finding the shortest path

For a given initial state, we can iteratively solve Equation 3.17 and compute the expected collision resolution lengths  $L^c(N, n)$  and the optimal control actions  $u^*$  corresponding to the possible states

### Numerical Results

For illustration purposes, consider  $N = 8$  and  $N = 16$  users. For  $N = 8$ , the number of active users at the beginning of an epoch may vary from 1 to 8. The various possible states are given in Table 3.1. The possible initial states will be  $(8,1), (8,2), \dots, (8,8)$ . We compute the state transition probabilities for all states using Equations 3.2 to 3.8. We then iterate according to Equation 3.17. The iteration is stopped when the following condition is satisfied

$$|V_{m+1}(S_i) - V_m(S_i)| < 1 \times 10^{-3} \quad \forall S_i \in \mathcal{S}$$

Thus we get the values of expected collision resolution lengths correct up to three decimal places. Table 3.1 gives the expected collision resolution length and the corresponding optimal control action for each possible state for  $N = 8$ . Similarly, for  $N = 16$ , the optimal control actions for various states are given in Table 3.2. In Table 3.2, those states which have already been included in Table 3.1 are not shown. The expected collision resolution lengths for all possible initial states when the total number of users is 8 and 16, are shown explicitly in Table 3.3 and 3.4 respectively.

We note from Table 3.1 that when the number of active users at the beginning of a collision resolution epoch is 8, the optimal control policy leads to enabling each user separately. This is nothing but TDMA which is obviously the best way to transmit when all users have packets. For other initial states, the algorithm performs better than TDMA.

In actual practice, since the users know the initial state, they have perfect knowledge of the state at every stage by observing the feedback and the control action. The users can determine the optimal control action corresponding to the state from the look-up table. This action specifies the number of users allowed to transmit their packets. Corresponding to this number, there will be many possible sets of users that can be enabled. The users arrive at a particular set by using a pseudorandom number generator with the same seed. The enabled users transmit their packets, if any. This procedure is continued till all users are resolved.

### 3.5 Steady State Average Collision Resolution Length

In this section, we perform the steady state analysis of our collision resolution algorithm. The system operation can be divided into successive collision resolution epochs. We assume that packets arrive to each user according to a Bernoulli process. Let  $\sigma$  be the probability with which a user generates a packet in each slot. But since the user has only one buffer to store a packet, the remaining packets are assumed to be lost. At the beginning of a collision resolution epoch, all users know the number of active users. Alternatively, all users are enabled initially. If no or only one user is active, then the collision resolution epoch consists of only one slot. In the event of a collision, the users measure the collision multiplicity as in Section 3.1 and come to know of the number of active users at the beginning of the collision resolution epoch.

Let  $\ell_i$  denote the length of  $i$ th collision resolution epoch. Then the probability that a user has a packet at the beginning of  $(i + 1)$ th collision resolution epoch will be given by

$$p^{i+1} = 1 - (1 - \sigma)^{\ell_i} \quad (3.18)$$

Note that  $\ell_i$  is a random variable and hence  $p$  varies from epoch to epoch.

Assuming that steady state conditions are obtained, the above expression can be written as

$$\bar{p} = 1 - (1 - \sigma)^{\bar{L}} \quad (3.19)$$

where  $\bar{L}$  denotes the steady state average collision resolution length and  $\bar{p}$  the steady state probability of having a packet by a user. Let  $p_n$  denote the probability that  $n$  users are active out of  $N$  users i.e.,

$$p_n = \binom{N}{n} \bar{p}^n (1 - \bar{p})^{N-n} \quad (3.20)$$

Then  $\bar{L}$  can be computed by the following expression

$$\bar{L} = \sum_{n=0}^N L^c(N, n) p_n \quad (3.21)$$

The value of  $\bar{L}$  can be calculated iteratively. Let  $\bar{L}(k+1)$  be the value of  $\bar{L}$  at  $(k+1)$ th iteration. Then using Equations 3.19 to 3.21, we can write the following

$$\begin{aligned}\bar{L}(k+1) &= \sum_{n=0}^N L^c(N, n) \binom{N}{n} p(k)^n (1-p(k))^{N-n} \\ \text{where } p(k) &= 1 - (1-\sigma)^{\bar{L}(k)}\end{aligned}\quad (3.22)$$

To begin with, initially,  $\bar{L}(0)$  can be taken as the arithmetic mean, i.e.,  $\bar{L}(0) = \sum_{n=0}^N L^c(N, n)/N$ . The iteration is stopped when the following condition is satisfied

$$\begin{aligned}|\bar{L}(k+1) - \bar{L}(k)| &< \epsilon \\ |p(k+1) - p(k)| &< \epsilon\end{aligned}$$

where  $\epsilon$  is the specified value

To illustrate the above case, we take the number of users  $N = 16$ . Note that under the steady state operation, we have  $L^c(N, 0) = L^c(N, 1) = 1$ . The values of  $L^c(N, n)$  for  $n \geq 2$  are given in Table 3.4. For these values, we perform the iteration of Equation 3.22 for the different values of  $\sigma$  till the convergence is achieved up to three decimal places. The steady state average collision resolution lengths and the steady state probabilities of having a packet at the beginning of the epoch for different values of arrival probabilities  $\sigma$  are tabulated in Table 3.5.

Table 3.1 Optimal Control Values for Possible States for  $N = 8$ 

State	Optimal Control	Expected Collision Resolution Length
(8,1)	8	1 000
(8,2)	4	2 571
(8,3)	2	4 189
(8,4)	2	5 471
(8,5)	2	6 688
(8,6)	2	7 357
(8,7)	1	7 750
(8,8)	1	8 000
(7,1)	7	7 000
(7,2)	3	2 571
(7,3)	2	4 100
(7,4)	2	5 360
(7,5)	2	6 238
(7,6)	1	6 714
(7,7)	1	7 000
(6,1)	6	6 000
(6,2)	3	2 533
(6,3)	2	3 875
(6,4)	2	5 066
(6,5)	1	5 666
(6,6)	1	6 000
(5,1)	5	1 000
(5,2)	2	2 500
(5,3)	2	3 800
(5,4)	1	4 600
(5,5)	1	5 000
(4,1)	4	1 000
(4,2)	2	2 333
(4,3)	1	3 500
(4,4)	1	4 000
(3,1)	3	1 000
(3,2)	1	2 333
(3,3)	1	3 000
(2,1)	2	1 000
(2,2)	1	2 000
(1,1)	1	1 000

Table 3.2 Optimal Control Values for Possible States for  $N = 16$ 

State	Optimal Control	Expected Collision Resolution Length
(16,1)	16	1 000
(16,2)	8	2 733
(16,3)	5	4 603
(16,4)	4	6 326
(16,5)	3	7 985
(16,6)	3	9 504
(16,7)	2	10 825
(16,8)	2	12 080
(16,9)	2	13 256
(16,10)	2	14 222
(16,11)	2	14 839
(16,12)	2	15 235
(16,13)	2	15 510
(16,14)	2	15 716
(16,15)	2	15 875
(16,16)	1	16 000
(15,1)	15	1 000
(15,2)	7	2 733
(15,3)	5	4 567
(15,4)	4	6 293
(15,5)	3	7 864
(15,6)	2	9 358
(15,7)	2	10 646
(15,8)	2	11 858
(15,9)	2	12 941
(15,10)	2	13 696
(15,11)	2	14 158
(15,12)	2	14 469
(15,13)	2	14 695
(15,14)	1	14 866
(15,15)	1	15 000

*Continued-*

State	Optimal Control	Expected Collision Resolution Length
(14,1)	14	1 000
(14,2)	7	2 725
(14,3)	5	4 526
(14,4)	3	6 232
(14,5)	3	7 776
(14,6)	2	9 155
(14,7)	2	10 416
(14,8)	2	11 599
(14,9)	2	12 513
(14,10)	2	13 063
(14,11)	2	13 419
(14,12)	2	13 670
(14,13)	2	13 857
(14,14)	1	14 000
(13,1)	13	1 000
(13,2)	6	2 717
(13,3)	4	4 482
(13,4)	3	6 146
(13,5)	3	7 658
(13,6)	2	8 993
(13,7)	2	10 203
(13,8)	2	11 274
(13,9)	2	11 944
(13,10)	2	12 358
(13,11)	2	12 641
(13,12)	2	12 846
(13,13)	1	13 000

*Continued—*

State	Optimal Control	Expected Collision Resolution Length
(12,1)	12	1 000
(12,2)	6	2 697
(12,3)	4	4.420
(12,4)	3	6 041
(12,5)	2	7 491
(12,6)	2	8 757
(12,7)	2	9 955
(12,8)	2	10 793
(12,9)	2	11 284
(12,10)	2	11 606
(12,11)	2	11 833
(12,12)	1	12 000
(11,1)	11	1 000
(11,2)	5	2 690
(11,3)	3	4 397
(11,4)	3	5 953
(11,5)	2	7 351
(11,6)	2	8 563
(11,7)	2	9 595
(11,8)	2	10 191
(11,9)	2	10 563
(11,10)	1	10 818
(11,11)	1	11 000
(10,1)	10	1 000
(10,2)	5	2 666
(10,3)	3	4 341
(10,4)	2	5 837
(10,5)	2	7 106
(10,6)	2	8 322
(10,7)	2	9 071
(10,8)	2	9 511
(10,9)	3	9 800
(10,10)	1	10 000

*Continued—*

State	Optimal Control	Expected Collision Resolution Length
(9,1)	9	1 000
(9,2)	4	2 638
(9,3)	3	4 238
(9,4)	2	5 719
(9,5)	2	6 947
(9,6)	2	7 911
(9,7)	2	8 444
(9,8)	1	8 777
(9,9)	1	9 000

Table 3 3 Expected Collision Resolution Length for  $N = 8$

State	Expected Collision Resolution Length
(8,1)	1 000
(8,2)	2 571
(8,3)	4 189
(8,4)	5 471
(8,5)	6 888
(8,6)	7 357
(8,7)	7.750
(8,8)	8 000

Table 3 4 Expected Collision Resolution Length for  $N = 16$ 

State	Expected Collision Resolution Length
(16,1)	1 000
(16,2)	2 733
(16,3)	4 603
(16,4)	6 326
(16,5)	7 985
(16,6)	9 504
(16,7)	10 825
(16,8)	12 080
(16,9)	13 256
(16,10)	14 222
(16,11)	14 839
(16,12)	15 235
(16,13)	15 510
(16,14)	15 716
(16,15)	15 875
(16,16)	16 000

Table 3 5 Steady State Average Collision Resolution Length for  $N = 16$ 

Arrival Probability $\sigma$	Steady State Probability $\bar{p}$	Average Collision Resolution Length $\bar{L}$
0 01	0 01	1 024
0 02	0 022	1 095
0 03	0 037	1 253
0 04	0 07	1 774
0 05	0 39	9 492
0 06	0 54	12 439
0 07	0 63	13 878
0 08	0 70	14 635
0 09	0 76	15 064
0 10	0 80	15 330
0 20	0 97	15 937
1 00	1 00	16 000

# Chapter 4

## Suboptimal Collision Resolution Algorithms

---

In Chapter 3, we have assumed the number of active users at the beginning of the collision resolution epoch to be known and derived an optimal collision resolution algorithm that minimizes the length of the collision resolution epoch for blocked channel access

Although the number of active users at the beginning of a collision resolution epoch can be known in practice through the mechanism explained in Section 3.1, we would like to consider a more general situation where the number of active users may not be known but the probability distribution over the initial states is given. If the packet arrival probability to each user is known, then the probability distribution over the initial states can be calculated

For such a general model, we pose the question *whether an optimal collision resolution algorithm can be derived*. We argue in this chapter that the probability distribution over the states conditioned on the past information constitutes the appropriate new state description for this problem. The problem of optimal collision resolution algorithm may then be viewed conceptually as the first passage problem of a Partially Observable Markovian Decision Process (POMDP). The optimal collision resolution algorithm can be derived in principle but the implementation of such an algorithm is computationally burdensome and therefore we look for *reasonably good* suboptimal algorithms which can be easily implemented in practice

We suggest in this chapter two suboptimal collision resolution algorithms. One of these suboptimal algorithms—SOA1 chooses control actions based on the state having maximum probability of occurrence while another suboptimal algorithm—SOA2 chooses control actions based on the expected state. The performance of these algorithms is studied by simulations.

The expected collision resolution lengths for various probability distributions due to the two suboptimal algorithms is compared with corresponding bounds on the optimal expected collision resolution lengths. We show that the degradation in performance due to these suboptimal algorithms is not significant for practical purposes.

Finally, we study the behavior of these two algorithms in the presence of feedback error. We discuss a feedback error model due to Massey and perform the simulations of our suboptimal algorithms under this model. The results of these simulations are presented in the end.

## 4.1 General Finite User Model

### 4.1.1 Basic Assumptions

Most of our assumptions are the same as those of Chapter 3. Following the notations of Chapter 3,  $N$  represents the total number of users in the system. The packets generated in one collision resolution epoch are transmitted in the next epoch. As before, we assume that a user can generate only one packet during a epoch. In other words, we may assume that each user has only one buffer to store a new packet during the epoch.

The users can start the transmission of their packets only at the slot beginnings. We call the slot containing only one packet as the successful slot and more than one packet as the collision slot. At the end of each slot, the users receive ternary feedback 0, 1 or 2 depending upon whether the slot was empty, successful or collision slot. Let the slot beginnings within a collision resolution epoch be denoted by  $t$  where  $t \in I$  and  $I = \{0, 1, 2, \dots\}$ .

Let  $N(t)$  and  $n(t)$  be the number of unresolved users and the total number

of active users, respectively, at time  $t$ . Then the state of our system at the time  $t$ , as defined in Chapter 3 and denoted by  $S(t)$ , is given to be two-tuple  $[N(t), n(t)]$ , i.e.,  $S(t) = [N(t), n(t)]$ . Clearly  $N(0)$  which is the number of unresolved users at the beginning of an epoch is equal to  $N$ . Similarly let  $n(0)$ , which is the number of active users at the beginning of the epoch, be equal to  $n$ . However, unlike the assumption made in Chapter 3, here the number of active users at the beginning of the epoch is *not known*.

We assume that, instead of the number of active users being known, the probability distribution over the initial states is given. If  $N$  is the total number of users in the system, then the number of possible states will be  $\mathcal{L} = N(N + 1)/2 + 1$ . Let the state space be  $\mathcal{S} = \{S_1, S_2, \dots, S_{\mathcal{L}}\}$ . The elements of the state space are denoted by  $S_i$  where  $S_i = (N_i, n_i)$  and  $N_i \leq N$  and  $n_i \leq N_i$ .

Let the initial probability distribution vector be denoted by  $P(0)$ , where  $P(0)$  is given by

$$P(0) = [p_{s_1}(0), p_{s_2}(0), \dots, p_{s_{\mathcal{L}}}(0)]$$

Here  $p_{s_1}(0) = \text{Prob}[S(0) = S_1]$ , i.e.,  $p_{s_1}(0)$  is the probability that the state at the beginning of the epoch is some two-tuple  $S_1 = (N_1, n_1)$ .

As in Chapter 3,  $\mathcal{U}(S_i)$  denote the set of admissible control actions, when the state is  $S_i$ . The control action specifies the number of users enabled for transmission. When a control action is employed at the instant  $t$ , the state  $S(t) = S_i$  moves to the state  $S(t + 1) = S_j$  with probability  $p_{S_i, S_j}(u)$  and a feedback  $\theta(t + 1)$  is observed by all users. The feedback  $\theta(t + 1)$  belongs to an observation space  $\Theta = \{0, 1, 2\}$ .

### 4.1.2 Dynamic Evolution of the System

With above assumptions, our general random access model may then be represented as shown in Figure 4.1. The feedback output  $\theta(t)$  depends probabilistically on  $S(t - 1)$  and  $u(t - 1)$ . Similarly, the state  $S(t) = [N(t), n(t)]$  depends probabilistically on  $S(t - 1)$  and  $u(t - 1)$ . Thus our system model is essentially a *Finite Probabilistic System* (FPS) introduced by Platzman [57].

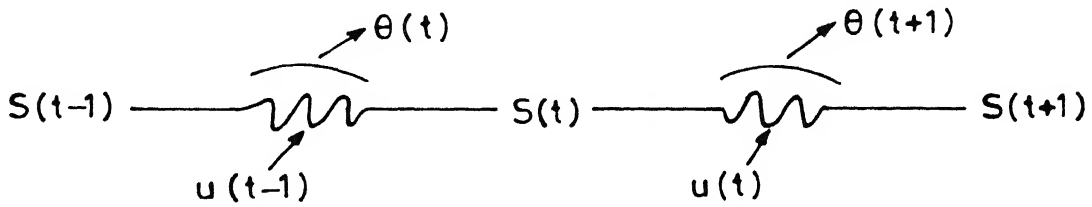


Figure 4.1 Representation of Finite User Random Access Model

Since both the state  $S(t) = [N(t), n(t)]$  and the feedback observation  $\theta(t)$  depend probabilistically on  $S(t-1)$  and  $u(t-1)$ , the pair  $[S(t), \theta(t)]$  is a random variable which depends jointly on  $[S(t-1), u(t-1)]$ . Thus the dynamic evolution of the system can be described by the probabilities of the state and the feedback values conditioned on the previous state and the control

As shown by Platzman [57], FPS is a generalization of Markov Chain and is equivalent to a Markov Decision Process (MDP) whose state at time  $t$  consists of the pair  $[S(t), \theta(t)]$ . Such a process is called a Partially Observable Markovian Decision Process.

Similar to FPS, the dynamic evolution of our system can be summarized in the following terms

- 1 The stochastic process consisting of the sequence of states  $\{S(t) = [N(t), n(t)], t \in I\}$  is called the core process. For a given sequence of control actions, it is a finite state Markov Chain. This chain is completely characterized by the state transition probability matrix  $P(u) = [p_{S_i, S_j}(u)]$ . The state transition probabilities can be calculated from Section 3.3.4.
- 2 The core process  $\{S(t), t \in I\}$  is not directly observable. The initial state  $S(0)$  assumes value  $S_i = (N_i, n_i)$  with probability  $p_{S_i}(0)$ .
- 3 The stochastic process  $\{\theta(t), t \in I\}$  is called the feedback process.
- 4 When the control  $u(t) = u$  is applied at time  $t$ , i.e., when  $u$  users are enabled at time  $t$ , the state  $S(t) = S_i$  undergoes a transition to the state  $S(t+1) = S_j$  with probability  $p_{S_i, S_j}(u)$  and feedback  $\theta(t+1) = \theta$  with

$\theta \in \{0, 1, 2\}$  is observed with probability  $p(\theta \mid S_i, u)$

### 4.1.3 The Conditional Probability Distribution of the State

The information vector at any instant  $t$  is defined as the data available to all users for making decision at the instant  $t$ . This data consists of all the past feedback observations and the control actions. Let this vector be denoted by  $I(t)$ . Then  $I(t)$  is given by

$$I(t) = \{\theta(1), \theta(2), \dots, \theta(t), u(0), u(1), \dots, u(t-1)\}$$

The following result can then be established

**Theorem 4.1** *The conditional probability of the state can be recursively computed by the following expression*

$$p[S(t+1) = S_r \mid I(t+1)] = \frac{\sum_{S_i \in S_\theta} p_{S_i, S_r}(u) p[S(t) \mid I(t)]}{\sum_{S_j \in S} \sum_{S_i \in S_\theta} p_{S_i, S_j}(u) p[S(t) \mid I(t)]} \quad (4.1)$$

where  $S_\theta$  is the set of all states  $S_i$  such that

$$p[\theta(t+1) = \theta \mid S(t) = S_i, S(t+1) = S_r, u(t) = u] = 1$$

and  $p_{S_i, S_j}(u) = p[S(t+1) = S_j \mid S(t) = S_i, u(t) = u]$

*Proof* We note that  $I(t+1) = \{\theta(1), \theta(2), \dots, \theta(t+1), u(0), u(1), \dots, u(t)\}$ . We then have

$$p[S(t+1) = S_r \mid I(t+1)] = p[S(t+1) = S_r \mid I(t), \theta(t+1), u(t)] \quad (4.2)$$

By applying Bayes' rule we can write

$$\begin{aligned} & p[S(t+1) = S_r \mid I(t+1)] \\ &= \frac{p[\theta(t+1) \mid S(t+1), I(t), u(t)] p[S(t+1) \mid I(t), u(t)]}{p[\theta(t+1) \mid I(t), u(t)]} \\ &= \frac{\sum_{S_i \in S} p_{S_i, S_r}(u) p[S(t) \mid I(t)] p[\theta(t+1) \mid S(t+1) = S_r, I(t), u(t)]}{\sum_{S_i \in S} \sum_{S_j \in S} p_{S_i, S_j}(u) p[S(t) \mid I(t)] p[\theta(t+1) \mid S(t+1) = S_j, I(t), u(t)]} \end{aligned} \quad (4.3)$$

where  $p_{S_i, S_r}(u) = p[S(t+1) = S_r | S(t) = S_i, u(t) = u]$  Now consider the term  $p[\theta(t+1) = \theta | S(t+1) = S_j, I(t), u(t)]$  This can be written as

$$\begin{aligned}
 & p[\theta(t+1) = \theta | S(t+1) = S_j, I(t), u(t)] \\
 &= \sum_{S_i \in \mathcal{S}} p[\theta(t+1) = \theta | S(t+1) = S_j, S(t) = S_i, I(t), u(t)] \\
 &\quad p[S(t) = S_i | S(t+1) = S_j, I(t), u(t)] \\
 &= \sum_{S_i \in \mathcal{S}} p[\theta(t+1) = \theta | S(t+1) = S_j, S(t) = S_i, u(t)] \\
 &\quad p[S(t) = S_i | S(t+1) = S_j, I(t), u(t)] \tag{4.4}
 \end{aligned}$$

The last equation follows from that fact that the feedback output depends only upon  $S(t)$ ,  $S(t+1)$  and  $u(t)$

The probability  $p[\theta(t+1) = \theta | S(t+1), S(t), u(t)]$  will assume values of either 0 or 1 for particular values of  $S(t+1)$ ,  $S(t)$  and  $u(t)$

Let us consider those values of  $S(t) = S_i$  such that

$$p[\theta(t+1) = \theta | S(t+1) = S_j, S(t) = S_i, u(t)] = 1 \tag{4.5}$$

Thus Equation 4.4 can be written as

$$\begin{aligned}
 & p[\theta(t+1) = \theta | S(t+1) = S_j, I(t), u(t)] = \\
 &\quad \sum_{S_i \in \mathcal{S}_\theta} p[S(t) = S_i | S(t+1) = S_j, I(t), u(t)] \tag{4.6}
 \end{aligned}$$

After some manipulations and applying Bayes' rule we get

$$p[\theta(t+1) = \theta | S(t+1) = S_j, I(t), u(t)] = \frac{\sum_{S_i \in \mathcal{S}_\theta} p_{S_i, S_j}(u) p[S(t) = S_i | I(t)]}{\sum_{S_i \in \mathcal{S}} p_{S_i, S_j}(u) p[S(t) = S_i | I(t)]} \tag{4.7}$$

Substituting this relation in Equation 4.3 we obtain

$$p[S(t+1) = S_r | I(t+1)] = \frac{\sum_{S_i \in \mathcal{S}_\theta} p_{S_i, S_r}(u) p[S(t) = S_i | I(t)]}{\sum_{S_j \in \mathcal{S}} \sum_{S_i \in \mathcal{S}_\theta} p_{S_i, S_j}(u) p[S(t) = S_i | I(t)]} \tag{4.8}$$

This establishes the result □

Let  $P(t) = \{p[S(t) = S_1 | I(t)], p[S(t) = S_2 | I(t)], \dots, p[S(t) = S_C | I(t)]\}$

Then Equation 4.8 can be written as function  $\mathcal{F}$  of  $P(t)$ ,  $u(t)$  and  $\theta(t + 1)$

$$P(t + 1) = \mathcal{F}\{P(t), u(t), \theta(t + 1)\} \quad (4.9)$$

Thus  $P(t + 1)$  depends only upon  $P(t)$ , the control action  $u(t)$  at time  $t$  and the observed feedback  $\theta(t + 1)$

We thus have an alternative representation of the finite user random access model where the state of the system is  $P(t)$ . The control policy enables certain users and due to the transmission of packets by these enabled users, a feedback  $\theta(t + 1)$  is observed and the state of the system  $P(t)$  moves to the next state  $P(t + 1)$

**Remark 4.1.1** Note that the state  $P(t + 1)$  depends only upon the previous state  $P(t)$  and the control action  $u(t)$ . Thus for a given sequence of control actions, the sequence  $\{P(t + 1), t \in I\}$  is a Markov Process

## 4.2 The Optimal Collision Resolution Problem

At the beginning of a collision resolution epoch, some  $n$  users may be active out of the total number of  $N$  users. The number of active users is, however, not known exactly, instead the initial probability vector  $P(0)$  is given. The collision resolution epoch ends when all users have resolved their collisions.

If we take  $P(t)$  as the state of our dynamical system, then the evolution of this state is completely governed by Equation 4.8. Thus given  $P(t)$ ,  $u(t)$  and  $\theta(t + 1)$ ,  $P(t + 1)$  can be calculated exactly. In other words, our model with  $P(t)$  as the state, will have perfect knowledge of the state at every stage. Let  $\tau$  be the time instant at which users have resolved collisions, then at this stage we reach the state  $P(\tau)$  whose all components are zero except the probability  $p[S(\tau) = (0, 0) | I(\tau)]$  which is equal to one. The problem of optimally resolving the collisions is that of finding the control policy (if it exists) which minimizes the expected time to reach the state where  $p[S(\tau) = (0, 0) | I(\tau)] = 1$ . The

optimal collision resolution problem may thus be viewed as the optimal first passage problem of a dynamical system with  $P(t)$  as the state. Similar to that of Chapter 3, the optimal collision resolution algorithm can be determined in principle. In Chapter 3, we had finite state space and the optimal control policy can be specified by a look-up table where corresponding to each state we have the control action. In this case, however, with  $P(t)$  as the state, we have an enlarged state space and the implementation of such an algorithm will be computationally burdensome.

We therefore do not go further into the discussion of the derivation of an optimal collision resolution algorithm. Instead of attempting to determine an optimal algorithm by computationally feasible means, we suggest two suboptimal algorithms. By comparing the performance of these two algorithms with bounds on the optimum performance, we conclude that these suboptimal collision resolution algorithms are quite satisfactory for practical purposes.

In the following section, we introduce these suboptimal algorithms. The performance of these algorithms is discussed in the next section.

## 4.3 Suboptimal Collision Resolution Algorithm

### 4.3.1 Suboptimal Collision Resolution Algorithm-1 (SOA1)

The suboptimal algorithm suggested here is based upon updating the conditional probability vector  $P(t)$  at every stage with the help of observed feedback and the control applied. The state  $S(t)$  is a two-dimensional vector  $[N(t), n(t)]$  and can assume any of the  $\mathcal{L}$  values from the state space  $\mathcal{S} = \{S_1, S_2, \dots, S_{\mathcal{L}}\}$  where  $S_i = (N_i, n_i)$  and  $N_i \leq N$  and  $n_i \leq N_i$ . The conditional probability vector gives the probability that the state  $S(t)$  can take each of these values at the instant  $t$ . We know that if the initial state is given, then the state at every stage can be known exactly. For such a perfect state information case, the optimal collision resolution algorithm is already known from Chapter 3.

In the suboptimal algorithm considered here (abbreviated as SOA1), the

users update the conditional probability vector and then choose the state which has the maximum value of probability. Assuming this state to be the actual state, the corresponding control is then applied. This control specifies the number of users to be enabled. For this number, there will be many possible sets of users that can be enabled. Specifically, if the number of unresolved users is  $N_i$  and the control action is  $u$ , then there will be total  $\binom{N_i}{u}$  possible combinations. The users choose one such set randomly by using a pseudorandom number generator with the same seed. Thus all users generate the same set at its site. The enabled users transmit their packets, if any, in the subsequent slot. The feedback due to the transmission of these  $u$  users is observed by all users. The users then update the conditional probability vector. This procedure is continued till all users have been resolved.

### Steps of SOA1

The steps of the algorithm can be summarized as

- Choose the state  $S_k = (N_k, n_k)$  which has the maximum value of probability
- Choose the control  $u_k$  corresponding to  $S_k$ .  $u_k$  would be the optimal value if  $S_k$  were the actual state
- The algorithm enables  $u_k$  number of users in the slot  $(t, t + 1)$
- Observe the feedback  $\theta(t + 1)$
- Update the conditional probability vector  $P(t + 1)$  by making use of the following equation

$$p[S(t + 1) = S_i | I(t + 1)] = \frac{\sum_{S_i \in S_\theta} p_{S_i, S_r}(u) p[S(t) = S_i | I(t)]}{\sum_{S_j} \sum_{S_i \in S_\theta} p_{S_i, S_j}(u) p[S(t) = S_j | I(t)]}$$

$$\forall S_r = (N_r, n_r) \in \mathcal{S}$$

- If  $p[S(t+1) = (0,0) | I(t+1)] = 1$ , then stop Else repeat from the starting step

### 4.3.2 Suboptimal Collision Resolution Algorithm–2 (SOA2)

From optimal control theory, we know that in Linear Quadratic Gaussian (LQG) problem, the certainty equivalent principle holds good and certainty equivalent control is equivalent to the optimal control. The second suboptimal algorithm considered in this section is motivated by this fact.

In the suboptimal algorithm considered here, we calculate the conditional probability vector and then compute the expected state. The conditional probability vector gives the probability that the state  $S(t)$  can take each of the  $\mathcal{L}$  values at the instant  $t$ . Let the expected state at the instant  $t$  be  $S_e = (N_e, n_e)$ . Each of the components of the expected state can be calculated as

$$N_e = \sum_{i=1}^{\mathcal{L}} N_i p[S(t) = S_i | I(t)] \quad (4.10)$$

$$n_e = \sum_{i=1}^{\mathcal{L}} n_i p[S(t) = S_i | I(t)] \quad (4.11)$$

We note here that  $N_e$  and  $n_e$  can take real values and the expected state  $S_e$  therefore may not belong to the state space  $\mathcal{S}$ . In such cases, the nearest integer values of  $N_e$  and  $n_e$  are taken. Denote this state by  $S_e^+ = (N_e^+, n_e^+)$ . The optimal control corresponding to this state  $S_e^+$  is then chosen assuming this to be the actual state. Let this control be denoted by  $u_e$ .

This control specifies the number of users to be enabled. For this number  $u_e$ , the users choose the particular set of users for transmission in a manner similar to that of SOA1. The enabled users transmit their packets and the corresponding feedback is received by all users. The users then update the conditional probability vector and the whole procedure is repeated till all users are resolved, i.e., we reach a stage where  $p[S(t) = (0,0) | I(t)] = 1$ .

#### Steps of SOA2

The steps of the algorithm SOA2 can be summarized as

- Calculate the probability vector  $P(t) = \{p[S(t) = S_1 | I(t)], p[S(t) = S_2 | I(t)], \dots, p[S(t) = S_L | I(t)]\}$
- Compute the expected state  $S_e = (N_e, n_e)$  where  $N_e = \sum_{i=1}^L N_i p[S(t) = (N_i, n_i) | I(t)]$  and  $n_e = \sum_{i=1}^L n_i p[S(t) = (N_i, n_i) | I(t)]$
- Compute the state  $S_e^+ = (N_e^+, n_e^+)$  where  $X^+$  denotes the nearest integer of  $X$
- Choose the optimal control  $u_e$  corresponding to  $S_e^+$
- Enable  $u_e$  number of users in the slot  $(t, t + 1)$
- Observe the feedback  $\theta(t + 1)$
- Update the conditional probability vector  $P(t + 1)$  by making use of Equation 4.8
- If  $p[S(t + 1) = (0, 0) | I(t + 1)] = 1$  then stop Else repeat from the beginning

## 4.4 Simulation of Suboptimal Collision Resolution Algorithms

In the previous section, we have introduced the two suboptimal algorithms. In this section, we discuss the performance of these algorithms. For illustration purposes, we perform the simulations for various initial probability distribution of the states for a total population of  $N = 8$  users. Since all users are unresolved at the beginning of a collision resolution epoch, initial probabilities of the states like  $(N_i, n_i)$  where  $N_i \neq 8$  are zero. The remaining probabilities are chosen arbitrarily.

Alternatively, if we know the probability  $\sigma$  with which a user generates a packet then since the user has only one buffer to store any new packet, it can have at-most one packet at the beginning of any epoch. Let  $p$  be the probability that a user has a packet at the beginning of the epoch, when the length of the previous epoch is  $\ell$ , then

$$p = 1 - (1 - \sigma)^\ell$$

In such cases, we may enable all users initially. If no user is active, then the users receive an empty feedback and the collision resolution epoch ends. If only one user is active, a success feedback is received and all users are resolved. If, however, a collision feedback is received then it is known that more than one user is active. Let  $p(n)$  denote the probability that  $n$  users are active given that at-least two users are active, then  $p(n)$  is given by,

$$p(n) = \frac{\binom{N}{n} p^n (1-p)^{N-n}}{1 - (1-p)^N - Np(1-p)^{N-1}} \quad \text{for } n \geq 2$$

The initial probabilities of the states of type  $(N, n)$  where  $N = 8$  and  $n$  varies from 2 to 8 can be calculated by the following relation,

$$p[S(0) = (N, n)] = p(n)$$

Our suboptimal algorithm can then be employed to resolve the collision.

#### 4.4.1 Procedure for Simulation

In all the examples discussed below, we perform the simulation experiment by taking each possible state (i.e., the state which has probability greater than zero) as the initial state and obtain the average collision resolution length for each case.

For each possible state, there will be several possible ways in which active users can be distributed amongst all users. If the state is  $(N, n)$ , then this distribution of  $n$  active users amongst total  $N$  users can be represented by an  $N$ -dimensional binary vector. In this vector, there will be  $n$  ones and  $(N - n)$  zeroes. The one in the  $k$ th position of this vector indicates that the user with the address  $k$  has a packet while zero indicates that the user does not have a packet. We call a particular realization of the users in this manner as an *Active*

Table 4 1 Active User Patterns

Address of User	Active User Pattern				
1	1	1	1	0	0
2	0	0	0	0	1
3	0	0	0	0	0
4	1	0	0	1	0
5	0	1	0	0	0
6	0	0	0	0	1
7	0	0	0	1	0
8	0	0	1	0	0

*User Pattern* For example, for the state  $(8,2)$ , there will be  $\binom{8}{2} = 28$  possible active user patterns. Some of them are shown in Table 4 1

In the first pattern of this example, we note that the users 1 and 4 have packets while rest other users do not have any packet. Similarly, we can explain for the other patterns. The simulation experiment is performed for each possible active user pattern and the average collision resolution length for a given initial state is obtained by taking averages over all simulation runs and all active user patterns of that state.

Note that the suboptimal algorithms SOA1 and SOA2 specify only the number of users to be enabled as the control action. For this control action, there will be many possible control vectors. Similar to a user pattern, a control vector is a binary vector with 1s and 0s. The one in a particular position of this vector indicates that the corresponding user is enabled for transmission. Clearly, the number of such ones is equal to the number of users to be enabled. The specific control vector is chosen randomly from amongst all possible vectors. For this, all users use a pseudorandom number generator using the same seed with the result that all users select the same vector. The enabled users transmit their packets if any.

For the same active user pattern, there can be several possible realizations of simulation experiments depending upon different control vectors used at each stage of the experiment. Sufficient number of simulations are performed

for each user pattern and the collision resolution lengths are averaged over all such simulation runs. The final expected collision resolution length is calculated by averaging the length over all user patterns. The number of runs are considered sufficient, if this expected collision resolution length remains unchanged to approximately three decimal places even if more number of runs are used.

In the following examples, the probability distributions are chosen to represent a range of user loading from light to heavy, but otherwise these probabilities are arbitrarily selected. Let  $\bar{L}_1$  and  $\bar{L}_2$  denote the expected collision resolution lengths for SOA1 and SOA2 respectively. Let  $\bar{L}_{bo}$  denote a bound (calculated as shown below) on the optimal expected collision resolution length.

**Example 4.4.1** The initial probabilities of the states for this example are given in Table 4.2. The probabilities of the remaining states are zero.

Table 4.2 Initial Probability Distribution-1

State	Probability
(8, 1)	0.90
(8, 2)	0.05
(8, 3)	0.05
(8, 4)	0.00
(8, 5)	0.00
(8, 6)	0.00
(8, 7)	0.00
(8, 8)	0.00

As explained earlier, the simulations were performed for every user pattern of each possible initial state, *i.e.*, of the states having nonzero probability, for the two suboptimal collision resolution algorithms— SOA1 and SOA2. A Total 3000 runs were found sufficient for each user pattern for the expected collision resolution length to converge to at-least three decimal places. The average collision resolution lengths for the three possible states, *i.e.*, (8, 1), (8, 2), (8, 3) obtained after averaging over 3000 runs and over all possible patterns are

tabulated in Table 4.3 for both the suboptimal algorithms

Table 4.3 Simulation Results for Probability Distribution-1

State	Average Collision Resolution Length	
	SOA1	SOA2
(8,1)	1 00000	1 00000
(8,2)	3 94527	4 47428
(8,3)	5 92335	5 64090

The states (8,1), (8,2) and (8,3) occur with probabilities 0.9, 0.05 and 0.05 respectively, hence the expected collision resolution lengths due to the suboptimal algorithms are given by,

$$\text{For SOA1} \quad \bar{L}_1 = 1.393$$

$$\text{For SOA2} \quad \bar{L}_2 = 1.405$$

*Bound on the Optimal Performance* The optimal collision resolution algorithm is known when the initial state is given. The optimal collision resolution lengths for each initial state are given in Table 4.4. The expected collision

Table 4.4 Expected Collision Resolution Length with Known Initial State

Initial State	Expected Collision Resolution Length
(8,1)	1 000
(8,2)	2 571
(8,3)	4 189
(8,4)	5 471
(8,5)	6 888
(8,6)	7 357
(8,7)	7 750
(8,8)	8 000

resolution length due to this algorithm when the initial state has the given

probability distribution of Table 4.2 can be shown to be equal to 1.238. This value would serve as a bound on the expected collision resolution length that would have been obtained if the optimal algorithm were known by following the formulation of Section 4.2.

We thus see here that the algorithm SOA1 performs marginally better than the algorithm SOA2. Both the algorithms perform worse than the optimal algorithm. While the expected collision resolution lengths due to the two suboptimal algorithms SOA1 and SOA2 are 1.393 and 1.405 respectively, the bound on the optimal expected collision resolution length is 1.238. Thus the degradation in performance for the two suboptimal algorithms is only about 12.5% and 13.4% respectively for this example.

**Example 4.4.2** The initial probabilities of the states for this example are chosen as indicated in Table 4.5.

Table 4.5 Initial State Probabilities–2

State	Probability
(8, 1)	0.70
(8, 2)	0.20
(8, 3)	0.05
(8, 4)	0.05
(8, 5)	0.00
(8, 6)	0.00
(8, 7)	0.00
(8, 8)	0.00

As in Example 4.4.1, the probabilities of other states are zero. In this case also, the simulations were performed for different runs for both the algorithms. The results obtained after 4000 simulation runs, which were considered adequate, are tabulated in Table 4.6.

The expected collision resolution lengths due to the two suboptimal algorithms are given by

$$\text{For SOA1} \quad \bar{L}_1 = 2.195$$

$$\text{For SOA2} \quad \bar{L}_2 = 2.260$$

Table 4.6 Simulation Results for Probability Distribution-2

State	Average Collision Resolution Length	
	SOA1	SOA2
(8, 1)	1 00000	1 00000
(8, 2)	3 72372	4 46107
(8, 3)	6 70293	6 00374
(8, 4)	8 32059	7 35400

*Bound on the Optimal Performance* For the probability distribution of this example,  $\bar{L}_{bo}$  is equal to 1 697

From these results, we again observe that SOA1 performs better than SOA2. While the bound on the optimal expected collision resolution length is 1 697, the expected collision resolution lengths due to the suboptimal algorithms SOA1 and SOA2 are 2 195 and 2 260 respectively. The performance degradation with respect to the bound is about 29.3% and 33% for the two algorithms.

**Example 4.4.3** For this example, we take the probability distribution as given in Table 4.7

Table 4.7 Initial State Probabilities-3

State	Probability
(8, 1)	0.10
(8, 2)	0.15
(8, 3)	0.25
(8, 4)	0.50
(8, 5)	0.00
(8, 6)	0.00
(8, 7)	0.00
(8, 8)	0.00

For this probability distribution, 7000 simulations runs for the algorithm SOA1 and 6000 runs for SOA2 were found sufficient. The simulation results are given in Table 4.8

Table 4.8 Simulation Results for Probability Distribution-3

State	Average Collision Resolution Length	
	SOA1 after 7000 runs	SOA2 after 6000 runs
(8,1)	3 50223	4 00000
(8,2)	4 71542	4 38934
(8,3)	5 53372	5 15610
(8,4)	5 75912	5 87807

The expected collision resolution lengths due to the two suboptimal algorithms are given by

$$\text{For SOA1} \quad \bar{L}_1 = 5\,320$$

$$\text{For SOA2} \quad \bar{L}_2 = 5\,286$$

The bound on the optimal expected collision resolution length in this case can be calculated to be equal to 4 268

We see that in this case the suboptimal algorithm SOA2 performs better than SOA1. The performance degradation due to SOA1 and SOA2 is about 24.6% and 23.8% respectively.

**Example 4.4.4** The initial probability distribution for this case is given in Table 4.9. This probability distribution can be considered as representative of heavy user loading.

In this case, the simulations were performed for 5000 runs. The average collision resolution lengths are tabulated in Table 4.10.

From these results we have

$$\text{For SOA1} \quad \bar{L}_1 = 8\,029$$

$$\text{For SOA2} \quad \bar{L}_2 = 8\,000$$

The bound on the optimal expected collision resolution length is readily seen to be equal to 7 887.

Table 4 9 Initial State Probabilities-4

State	Probability
(8, 1)	0 00
(8, 2)	0 00
(8, 3)	0 00
(8, 4)	0 00
(8, 5)	0 05
(8, 6)	0 05
(8, 7)	0 10
(8, 8)	0 80

Table 4 10 Simulation Results for Probability Distribution-4

State	Average Collision Resolution Length	
	SOA1	SOA2
(8, 5)	7 78479	8 0
(8, 6)	8 32013	8 0
(8, 7)	8 24602	8 0
(8, 8)	8 00000	8 0

For this example, SOA2 performs marginally better than SOA1. Note that both the suboptimal algorithms give performances which are close to the bound.

**Example 4.4.5** In this example, the initial probability distribution is given in Table 4 11.

The average collision resolution lengths for this probability distribution after 4000 simulation runs are tabulated in Table 4 12.

The expected collision resolution lengths can be calculated as

$$\text{For SOA1} \quad \bar{L}_1 = 7.780$$

$$\text{For SOA2} \quad \bar{L}_2 = 6.766$$

For this probability distribution  $\bar{L}_{bo} = 5.403$

Table 4 11 Initial State Probabilities-5

State	Probability
(8, 1)	0 125
(8, 2)	0 125
(8, 3)	0 125
(8, 4)	0 125
(8, 5)	0 125
(8, 6)	0 125
(8, 7)	0 125
(8, 8)	0 125

Table 4 12 Simualtion Results for Probability Distribution-5

State	Average Collision Resolution Length	
	SOA1	SOA2
(8, 1)	1 00000	3 24937
(8, 2)	4 07229	4 08930
(8, 3)	6 80261	5 42930
(8, 4)	8 64269	6 85850
(8, 5)	9 72966	7 97670
(8, 6)	10 25106	8 59441
(8, 7)	10 74631	8 93931
(8, 8)	11 00000	9 00000

In this example also, the suboptimal algorithm SOA2 is seen to perform better than SOA1. The degradation in performance with respect to the bound  $\bar{L}_{bo}$  for SOA1 and SOA2 is about 43% and 25 2% respectively

**Remark 4.4.1** We note that in Example 4 4 1 and Example 4.4 2, the suboptimal algorithm SOA1 which chooses control actions based on the state having maximum value of probability performs better than the algorithm SOA2 which chooses control actions based on the expected state. The initial probability distributions of these examples represent the cases when the users are lightly loaded. When the users are moderately and heavily loaded, as in other

examples, the algorithm SOA2 is seen to perform better

**Remark 4.4.2** The degradation in performance for the two suboptimal algorithms with respect to a bound is not significant for the examples considered. Since the optimal algorithm for the case when the initial state is known only probabilistically, will perform *worse* than this bound, it is reasonable to assert that these suboptimal algorithms are quite satisfactory

In the above examples, the initial probability distributions of the states were chosen to represent a range of user loading but otherwise arbitrary. If the packet arrival process to each user is a Bernoulli process with arrival probability  $\sigma$ , then as mentioned earlier, we may enable all users initially. If no user or only one user is active, then the collision resolution epoch ends. However, if collision occurs, then it is known that more than one user is active. Given the packet arrival probability  $\sigma$  and the fact that more than one user is active, we can then calculate the initial probability distribution and employ the collision resolution algorithms. The following two examples approximate the cases when the packet arrival probability  $\sigma$  is 0.1 and 0.3 respectively and where it is known that more than one user is active.

**Example 4.4.6** The probability distribution is given in Table 4.13. The simu-

Table 4.13 Initial State Probabilities-6

State	Probability
(8, 1)	0.000
(8, 2)	0.796
(8, 3)	0.176
(8, 4)	0.028
(8, 5)	0.000
(8, 6)	0.000
(8, 7)	0.000
(8, 8)	0.000

lation results after 7000 runs are given in Table 4.14

Table 4 14 Simulation Results for Probability Distribution-6

State	Average Collision Resolution Length	
	SOA1	SOA2
(8, 2)	2 68055	2 69557
(8, 3)	5 33706	5 29235
(8, 4)	8 09977	8 02116

The expected collision resolution lengths are

$$\text{For SOA1} \quad \bar{L}_1 = 3\,299$$

$$\text{For SOA2} \quad \bar{L}_2 = 3\,301$$

For this probability distribution  $\bar{L}_{opt} = 2\,936$

**Example 4.4.7** The probability distribution for this example is given in Table 4.15

Table 4 15 Initial State Probabilities-7

State	Probability
(8, 1)	0 000
(8, 2)	0 398
(8, 3)	0 341
(8, 4)	0 182
(8, 5)	0 062
(8, 6)	0 017
(8, 7)	0 000
(8, 8)	0 000

The simulations were performed for 4000 runs and the average collision lengths obtained after averaging over all these runs for each possible initial state are given in Table 4 16

The expected collision resolution lengths can be easily calculated as follows

$$\text{For SOA1} \quad \bar{L}_1 = 4\,972$$

Table 4.16 Simulation Results for Probability Distribution-7

State	Average Collision Resolution Length	
	SOA1	SOA2
(8, 2)	2 89908	3 78721
(8, 3)	5 11228	4 70663
(8, 4)	7 25946	6 45033
(8, 5)	9 19268	8 20135
(8, 6)	10 86720	10 06566

$$\text{For SOA2} \quad \bar{L}_2 = 4.965$$

For this probability distribution  $\bar{L}_{opt} = 3.999$

## 4.5 Performance of Suboptimal Algorithms in the presence of Feedback Error

In the previous sections, we have assumed that the feedback channel is noiseless and the users know the status of the slot correctly, *i.e.*, whether the slot is empty or it contains one packet or more than one packet. In a realistic situation, the users may observe the feedback incorrectly due to the presence of channel noise. Apart from the fact that the initial state is known only probabilistically, the feedback error introduces further uncertainty about the state of the system. In this section, we see how the feedback error affects the performance of our collision resolution algorithms. The model considered for this study was introduced by Massey [44]. We first describe this model and then study the performance of the algorithms under this channel model.

### Massey's Feedback Channel Model

The channel model introduced by Massey for feedback error analysis is the discrete memoryless channel as shown in Figure 4.2. The channel input is the actual status of the slot and the channel output is the feedback received by users. The model assumes that if a slot contains more than one packet, then it

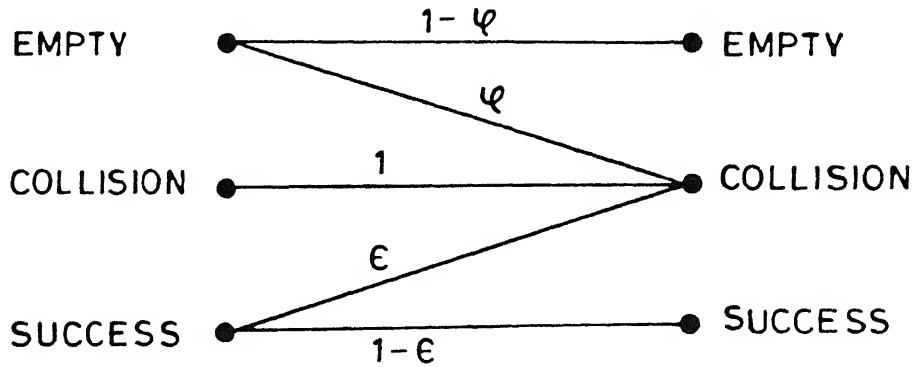


Figure 4.2 Massey's Model for Feedback Channel

would be correctly perceived by the users as the collision with probability one. If the slot is empty, then with probability  $\varphi$ , it could be interpreted by the users as the collision slot. An empty slot, however, will never be interpreted as the successful slot. The motivation behind assuming this is that in practice each packet would be encoded with some kind of error correcting code so that the probability that the receiver would identify an empty or a collision slot as a success would be very low.

Similarly in this model, a slot with successful transmission can be mistaken as a collision slot with probability  $\epsilon$ . It is realistic to assume that the only types of errors that can occur on the channel are those which result in the receiver interpreting the feedback as collision when the actual status of the slot in fact had been either empty or success.

Let  $\theta_e$  denote the actual feedback received by the users when the true feedback is  $\theta$ . Let 0, 1 and 2 represent empty, success and collision respectively. Then our model can be represented by the following relations:

$$\begin{aligned}
 p[\theta_e = 2 \mid \theta = 1] &= \epsilon, & p[\theta_e = 1 \mid \theta = 1] &= 1 - \epsilon, & p[\theta_e = 0 \mid \theta = 1] &= 0 \\
 p[\theta_e = 2 \mid \theta = 0] &= \varphi, & p[\theta_e = 1 \mid \theta = 0] &= 0, & p[\theta_e = 0 \mid \theta = 0] &= 1 - \varphi \\
 p[\theta_e = 2 \mid \theta = 2] &= 1, & p[\theta_e = 1 \mid \theta = 2] &= 0, & p[\theta_e = 0 \mid \theta = 2] &= 0
 \end{aligned}$$

With this channel model to represent feedback errors, we perform the simulations of our suboptimal collision resolution algorithms in the next section.

### 4.1 Simulation of Suboptimal Algorithms in the presence of Feedback Error

For illustration purposes, similar to Section 4.4, we take the number of users to be equal to 8. The simulation procedure adopted is also similar to that of the previous section. We perform the experiment for two initial probability distributions. For each possible initial state, there will be several ways in which the eight users are distributed among the total population. The similar experiment is performed for sufficient number of runs for every possible realization of the state and the average collision resolution length is obtained by averaging over all realizations and the simulation runs. Here, however, the users receive corrupted feedback information. In this experiment, the actual feedback information obtained due to this control action is passed through Massey's channel before being received by the users. The two examples to illustrate the effects of these channel errors are discussed below.

**Example 4.5.1** The initial probabilities for this example are chosen to be the same as that of Example 4.4.1 and given in Table 4.2. We have seen that the algorithm SOA1 performs better in this case. We choose this algorithm to study the effects of channel errors for this and the next example. The average collision resolution lengths obtained for each initial state after 4000 simulation runs are given in Table 4.17 for various values of error probabilities  $\epsilon$  and  $\varphi$ . The

Table 4.17 Results for Probability Distribution-1 Feedback Error

State	$\epsilon = 0.01$ $\varphi = 0.01$	$\epsilon = 0.10$ $\varphi = 0.01$	$\epsilon = 0.01$ $\varphi = 0.10$	$\epsilon = 0.50$ $\varphi = 0.01$	$\epsilon = 0.01$ $\varphi = 0.50$
(8, 1)	1.02806	1.29565	1.03025	3.08912	1.04990
(8, 2)	4.00031	4.41959	4.06241	6.52030	4.38991
(8, 3)	5.96319	6.18345	5.96439	7.76496	5.98151
$\bar{L}_1$	1.423	1.695	1.428	3.494	1.462

expected collision resolution length  $\bar{L}_1$  for this probability distribution is also shown in the table.

We see from this table that if  $\varphi = 0.01$  and  $\epsilon$  is varied as 0.01, 0.1 and 0.5, the expected collision resolution length  $\bar{L}_1$  varies as 1.423, 1.695 and 3.494. However, if  $\epsilon = 0.01$  and  $\varphi$  is varied as 0.01, 0.1 and 0.5, then  $\bar{L}_1$  varies as 1.423, 1.428 and 1.462 respectively. We thus note that the effect of  $\varphi$  is less pronounced than that of  $\epsilon$  on the performance of the algorithm.

**Example 4.5.2** The initial probability distribution for this example is the same as that of Example 4.4.2 (given in Table 4.5). The average collision resolution lengths for each state after sufficient simulation runs are given in Table 4.18 for various values of  $\epsilon$  and  $\varphi$ . The total expected collision resolution length is also calculated and shown in the table.

Table 4.18. Results for Probability Distribution-2 Feedback Error

State	$\epsilon = 0.01$ $\varphi = 0.01$	$\epsilon = 0.10$ $\varphi = 0.01$	$\epsilon = 0.01$ $\varphi = 0.10$	$\epsilon = 0.50$ $\varphi = 0.01$	$\epsilon = 0.01$ $\varphi = 0.50$
(8, 1)	1.02790	1.28712	1.03012	2.90653	1.04490
(8, 2)	3.76901	4.18202	3.81395	6.51599	4.11222
(8, 3)	6.75601	7.14088	6.79681	8.90598	7.00136
(8, 4)	8.34453	8.55260	8.34065	10.25330	8.33762
$\bar{L}_1$	2.227	2.251	2.240	4.295	2.320

Similar to the above example, we see that as  $\epsilon$  is varied as 0.01, 0.1 and 0.5 keeping  $\varphi$  fixed at 0.01, the expected collision resolution length  $\bar{L}_1$  varies as 2.227, 2.521 and 4.295. However, if  $\epsilon$  is fixed at 0.01 and  $\varphi$  varies as 0.01, 0.1 and 0.5, the value of  $\bar{L}_1$  varies 2.227, 2.240 and 2.320.

The important conclusion of these results is that the probability  $\epsilon$  which is the probability that a successful slot will be interpreted as a collision slot has marked effect on the performance while the probability  $\varphi$  which is the probability that an empty slot will be interpreted as a collision slot does not have significant effect.

## Chapter 5

# A Blocked Random Access Algorithm for Buffered Users

---

In the previous chapters, we have considered users each with a single packet. In any practical network, a user will have a buffer with a storage capacity of more than one packet. In this chapter, we consider the case of buffered users. For analytical simplicity, we assume that the buffer size is infinite, although in practice a user can have only finite storage capacity. For such buffered users, we adopt and analyze an access protocol.

We begin this chapter by looking at the nature of the queueing problem for buffered users when a simple random access algorithm like ALOHA is used. We then discuss the system model and the mechanism of our protocol. The operation of the system can be divided into successive transmission intervals called transmission epochs. We show that the sequence of the transmission epoch lengths is a Markov chain. The expressions for evaluating the transition probabilities of this Markov chain are then derived. The ergodicity of this chain implies the stability of the system. Sufficient condition for the ergodicity is stated and proved. We suggest an iterative and truncation procedure to calculate the steady state probabilities of the transmission epoch length. By obtaining the steady state probabilities, the expected transmission epoch length under dynamic operating condition can be calculated. The average packet delay analysis is then performed. It is shown that the packet delay consists of three components. Expressions for each of these components are derived.

and their significance is explained. The expected transmission epoch lengths and packet delays are then compared with the corresponding quantities for a polling type of protocol suggested and analyzed here. This protocol is in fact a variant of TDMA for buffered users.

## 5.1 Queueing Problem for Buffered Users

When a random multiple access protocol is used for a finite population of buffered users, an interesting problem of interacting and coupled queues arises. The analysis of the steady state behavior of the system is in general difficult to evaluate. Even if a simple random access algorithm like ALOHA is used for buffered users, the behavior of the system is known only for some simple cases.

In general, the behavior of these systems can be modelled by an  $N$ -dimensional Markov Chain with the state defined as  $\mathbf{Q} = (Q_1, Q_2, \dots, Q_N)$  where  $Q_i$  is the number of packets in the queue of  $i$ th user and  $N$  is the total number of users in the system. The problem here is concerned with the enormous size of the state space. Szpankowski [71] has studied the ergodicity of such an  $N$ -dimensional Markov chain and has obtained upper and lower bounds for the stable region of the system. In [68], a system consisting of only two interfering queues is considered and exact analytical results are derived. For systems with queues greater than two, they have suggested an approximate method. Several other approximate models to study the behavior of general interacting queues have been considered [36, 70]. In [66], Saadawi and Ephremides have used two Markov chains, one describes the state of the system and the other, the state of a user. The average packet delay and throughput can then be calculated by solving a set of simultaneous coupled nonlinear equations. Another approximate model that reduces the size of the state space has been proposed in [70].

In the protocol considered in this chapter, we use the optimal collision resolution algorithm of Chapter 3. But only the first packet of the buffer participates in the collision resolution process. The remaining packets present

the buffer at the beginning of the epoch follow a *Come Right-in* type of collision free strategy. The analysis of the protocol is then considerably simplified as compared to ALOHA for buffered users.

## 5.2 System Model

As mentioned in Chapter 3, we consider a finite population of  $N$  users. However, each user instead of having storage capacity of just one packet, is provided with a buffer of infinite size. All other assumptions regarding the system are the same as those of Chapter 3. The channel time, as before, is slotted. At the end of each slot, all users receive ternary feedback. The feedback is assumed to be errorless.

The packet generation process at each user is assumed to be a discrete time Bernoulli process. Let  $\sigma$  be the probability with which a user generates a packet in a slot. A user can generate at most one packet in each slot. We assume for convenience that a packet is generated at the beginning of a slot. The packets are stored in the buffer till they are transmitted successfully. We discuss below the mechanism of this access protocol. The first packet in the buffer of a user is referred as the head of the line (HOL) in the sequel.

### 5.2.1 Mechanism of Access Protocol

We consider the dynamic operation of the system. This dynamic operation of the system can be divided into successive transmission epochs as shown in Figure 5.1. Only those packets which have been generated in the previous epoch are transmitted in the current transmission epoch. The remaining packets, as and when they arrive, are queued in the buffer. Thus the access protocol is of the blocked access type.

**Definition 5.1** *We say that a packet in the buffer of any active user is an old packet if it was generated in the previous epoch.*

Thus all users which are active at the beginning of an epoch transmit their old packets in the current epoch. The access protocol consists of the collision

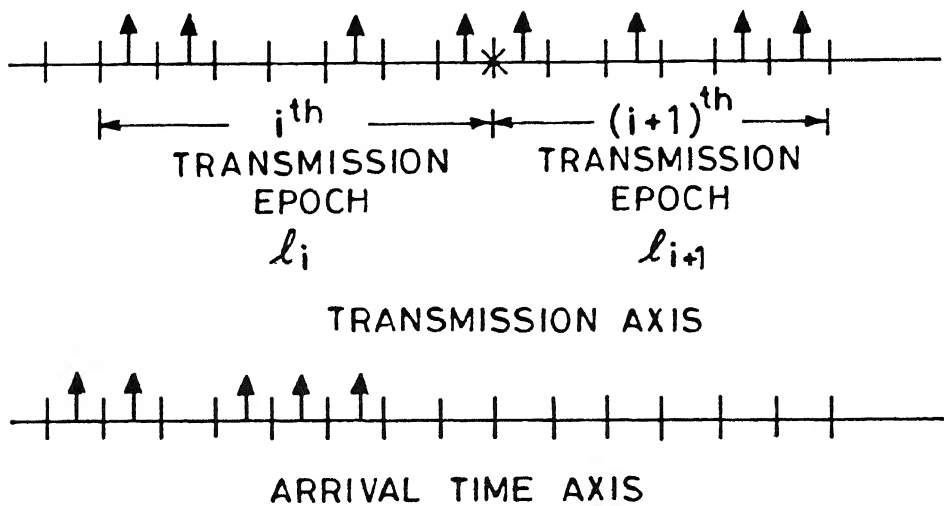


Figure 51 Dynamic Operation of the Protocol

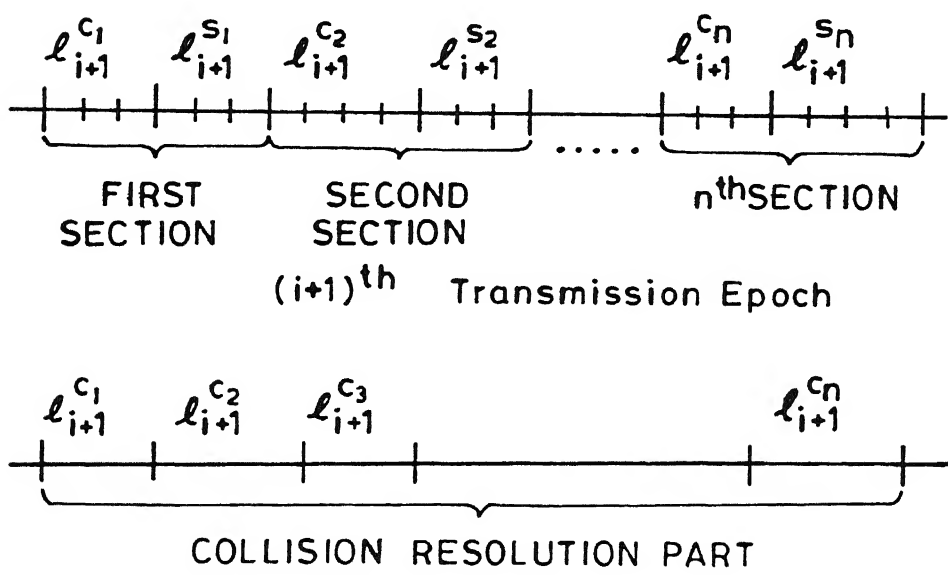


Figure 52 Transmission Epoch

esolution algorithm of Chapter 3 which is used by the HOL packet and a collision free strategy used by the other old packets of the buffer

At the beginning of a transmission epoch, the number of active users is known to each user. The mechanism of knowing this has already been explained in Section 3.1. The HOL of each user participates in the collision resolution process using the collision resolution algorithm. As soon as any of these packets is transmitted successfully, the collision resolution algorithm is interrupted and all other old packets of the successful user are transmitted in consecutive slots. An empty slot at the end indicates the end of the transmission of these packets. After this, the collision resolution algorithm starts again from the point where it was interrupted. This procedure continues till the active users have transmitted all their old packets.

Each transmission epoch as shown in Figure 5.2 can be seen to consist of several sections. If the number of active users at the beginning of an epoch is  $n$ , then there will be total  $n$ -sections in the epoch, each section corresponding to an active user. Let  $\ell_{i+1}$  be the length of  $(i + 1)$ th epoch. Each section has two subsections— one *Collision Resolution Subsection* and another *Successful Packet Transmission Subsection*. The collision resolution subsection ends with the successful transmission of HOL of an active user.

Let  $\ell_{i+1}^c$  and  $\ell_{i+1}^s$  denote the lengths of the collision resolution subsection and the successful packet transmission subsection of  $j$ th section of  $(i + 1)$ th epoch. The collision resolution part of an epoch is the sum of the collision resolution subsections of the epoch. Similarly the successful packet transmission part of an epoch is equal to the sum of the successful packet transmission subsections. Let the lengths of these parts be denoted by  $\ell_{i+1}^c$  and  $\ell_{i+1}^s$  respectively. We then have

$$\ell_{i+1} = \ell_{i+1}^c + \ell_{i+1}^s$$

and

$$\begin{aligned} \ell_{i+1}^c &= \ell_{i+1}^{c_1} + \ell_{i+1}^{c_2} + \dots + \ell_{i+1}^{c_n} \\ \ell_{i+1}^s &= \ell_{i+1}^{s_1} + \ell_{i+1}^{s_2} + \dots + \ell_{i+1}^{s_n} \end{aligned}$$

## 5.2.2 The System Markov Chain

Consider the sequence of transmission epoch lengths  $\{\ell_i, i = 0, 1, \dots\}$ . The length of any transmission epoch (say  $\ell_{i+1}$  of  $(i + 1)$ th epoch) is a random variable and depends upon the number of packets transmitted in it. Due to the nature of the protocol considered here, the number of packets transmitted in an epoch depend only upon the length of the previous epoch. This fact can be expressed as

$$Prob[\ell_{i+1} = k \mid \ell_i = j, \ell_{i-1}, \dots, \ell_0] = Prob[\ell_{i+1} = k \mid \ell_i = j]$$

In other words, we have the following lemma

**Lemma 5.1** *The sequence of transmission epoch lengths  $\{\ell_i, i = 0, 1, \dots\}$  is a Markov chain with state space  $\mathcal{Z} = \{1, 2, \dots\}$*

We will discuss the ergodicity of this Markov chain later. But first, we turn our attention to the evaluation of the probabilities of state transitions.

## 5.3 Evaluation of the State Transition Probabilities

Let  $\pi_{j,k}$  denote the probability of transition from the state  $j$  to the state  $k$  of the Markov chain  $\{\ell_i, i = 0, 1, \dots\}$ , i.e.,  $\pi_{j,k} = Prob[\ell_{i+1} = k \mid \ell_i = j]$ . These state transition probabilities can be evaluated as follows.

Let  $\eta_i$  denote the number of active users at the beginning of  $i$ th transmission epoch, then  $\pi_{j,k}$  can be written as

$$\begin{aligned} \pi_{j,k} &= Prob[\ell_{i+1} = k \mid \ell_i = j] \\ &= \sum_{n=0}^N Prob[\ell_{i+1} = k \mid \ell_i = j, \eta_{i+1} = n] Prob[\eta_{i+1} = n \mid \ell_i = j] \end{aligned} \quad (5.1)$$

where  $Prob[\eta_{i+1} = n \mid \ell_i = j] = p(n \mid j)$  denotes the probability that the number of active users at the beginning of a transmission epoch is  $n$  given that the

length of the previous epoch was  $j$ . If  $p$  is the probability that a user is active at the beginning of an epoch, then  $p(n | j)$  is given by

$$p(n | j) = \binom{N}{n} p^n (1-p)^{N-n} \quad (5.2)$$

Since each user generates a packet with probability  $\sigma$  in a slot and if the length of the previous epoch is  $j$ , we have

$$p = 1 - (1 - \sigma)^j \quad (5.3)$$

We then have from Equation 5.1

$$\begin{aligned} \pi_{j,k} &= \text{Prob}[\ell_{i+1} = k | \ell_i = j, \eta_{i+1} = 0] p(0 | j) \\ &\quad + \sum_{n=1}^N \text{Prob}[\ell_{i+1} = k | \ell_i = j, \eta_{i+1} = n] p(n | j) \end{aligned} \quad (5.4)$$

If no user is active at the beginning of an epoch, then the epoch will consist of one empty slot only. Hence

$$\text{Prob}[\ell_{i+1} = k | \ell_i = j, \eta_{i+1} = 0] = \begin{cases} 1, & \text{for } k = 1 \\ 0, & \text{otherwise} \end{cases} \quad (5.5)$$

$\pi_{j,k}$  can therefore be written as

$$\pi_{j,k} = \Delta_{0,k-1} p(0 | j) + \sum_{n=1}^N \text{Prob}[\ell_{i+1} = k | \ell_i = j, \eta_{i+1} = n] p(n | j) \quad (5.6)$$

where

$$\Delta_{j,k} = \begin{cases} 1, & \text{for } j = k \\ 0, & \text{otherwise} \end{cases}$$

Since  $\ell_{i+1} = \ell_{i+1}^c + \ell_{i+1}^s$  where the collision resolution part  $\ell_{i+1}^c$  and the successful packet transmission part  $\ell_{i+1}^s$  are independent random variables,  $\text{Prob}[\ell_{i+1} = k | \ell_i = j, \eta_{i+1} = n]$  is the convolution of the probabilities of  $\ell_{i+1}^c$  and  $\ell_{i+1}^s$ . Hence we have

$$\begin{aligned} \pi_{j,k} &= \Delta_{0,k-1} p(0 | j) + \sum_{n=1}^N \sum_{k_1=0}^{n_j} \text{Prob}[\ell_{i+1}^s = k_1 | \ell_i = j, \eta_{i+1} = n] \\ &\quad \text{Prob}[\ell_{i+1}^c = k - k_1 | \ell_i = j, \eta_{i+1} = n] p(n | j) \\ &= \Delta_{0,k-1} p(0 | j) + \sum_{n=1}^N \sum_{k_1=0}^{n_j} \pi_{j,n}^s(k_1) \pi_n^c(k - k_1) p(n | j) \end{aligned} \quad (5.7)$$

In this equation  $\pi_{j,n}^s(k_1) = \text{Prob}[\ell_{i+1}^s = k_1 \mid \ell_i = j, \eta_{i+1} = n]$  is the probability distribution of the successful packet transmission part of the epoch. Similarly  $\pi_n^c(k - k_1) = \text{Prob}[\ell_{i+1}^c = k - k_1 \mid \ell_i = j, \eta_{i+1} = n]$  is the probability distribution of the collision resolution part of the transmission epoch and is independent of the previous epoch length. We will determine  $\pi_{j,n}^s(k_1)$  and  $\pi_n^c(k - k_1)$ .

### 5.3.1 Determination of $\pi_{j,n}^s(k_1)$

Define  $C_{n,j}(k_1)$  as follows

If  $j = 1$  then

$$C_{n,1}(k_1) = \begin{cases} 1, & \text{if } k_1 = n \\ 0, & \text{otherwise} \end{cases} \quad (5.8)$$

If  $j > 1$  then

$$C_{n,j}(k_1) = \sum \binom{j}{k'_1} \binom{j}{k'_2} \binom{j}{k'_n} \quad (5.9)$$

The above summation is over the set  $K = \{(k'_1, k'_2, \dots, k'_n) \mid \sum_{i=1}^n k'_i = k_1, \forall k'_i \geq 1\}$

We then have the following theorem

**Theorem 5.1** *The probability distribution of the successful packet transmission part of the epoch  $\pi_{j,n}^s(k_1)$  is given by*

$$\pi_{j,n}^s(k_1) = \begin{cases} \frac{\sigma^{k_1}(1-\sigma)^{j^n-k_1}}{\{1-(1-\sigma)^j\}^n} C_{n,j}(k_1), & \text{for } n \leq k_1 \leq nj \\ 0, & \text{otherwise} \end{cases} \quad (5.10)$$

where  $C_{n,j}(k_1)$  is as defined above

*Proof* We consider three different cases

1 For  $n \geq 1$  and  $j = 1$

In this case the statement of the theorem reduces to the following

$$\pi_{j,n}^s(k_1) = \begin{cases} 1, & \text{for } k_1 = n \\ 0 & \text{otherwise} \end{cases} \quad (5.11)$$

Since the length of the previous epoch is one slot only, at most one packet could have arrived to each user. If the number of active users is  $n$ , each of them would contribute one slot to the successful packet transmission part of the epoch and thus the length of this part can only be  $n$ . This proves the statement.

2 For  $n = 1$  and  $j > 1$

If  $n = 1$  and  $j > 1$  then  $\pi_{j,n}^s(k_1)$ , according to the theorem, is given by

$$\pi_{j,n}^s(k_1) = \begin{cases} \frac{\binom{j}{k_1} \sigma^{k_1} (1-\sigma)^{j-k_1}}{1 - (1-\sigma)^j}, & \text{for } 1 \leq k_1 \leq j \\ 0, & \text{otherwise} \end{cases} \quad (5.12)$$

If only one user is active, then the length of the successful packet transmission part is equal to the number of old packets in the buffer other than the HOL packet plus one slot accounting for an empty one indicating the termination of packets' transmissions. This is equal to the number of old packets in the buffer given that the user contains at-least one packet. Since a user generates a packet with probability  $\sigma$ , we have

$$\begin{aligned} Prob[\text{Number of old packets in the buffer of the active user} = k_1] \\ = \frac{\binom{j}{k_1} \sigma^{k_1} (1-\sigma)^{j-k_1}}{1 - (1-\sigma)^j} \quad \text{for } 1 \leq k_1 \leq j \end{aligned} \quad (5.13)$$

$\pi_{j,n}^s(k_1)$  for  $n = 1$  and  $j > 1$  is then given by

$$\pi_{j,n}^s(k_1) = \begin{cases} \frac{\binom{j}{k_1} \sigma^{k_1} (1-\sigma)^{j-k_1}}{1 - (1-\sigma)^j}, & \text{for } 1 \leq k_1 \leq j \\ 0, & \text{otherwise} \end{cases} \quad (5.14)$$

3 For  $n > 1$  and  $j > 1$

Let  $a_1, a_2, \dots, a_n$  be the addresses of  $n$  active users for a specific realization of active user pattern. Let  $b_{i+1}^{a_r}$  be the number of packets in the buffer of the user  $a_r$  at the beginning of  $(i+1)$ th epoch. Then

$$Prob[b_{i+1}^{a_r} = k'_r \mid \ell_i = j] = \frac{\binom{j}{k'_r} \sigma^{k'_r} (1-\sigma)^{j-k'_r}}{1 - (1-\sigma)^j} = \xi(k'_r) \quad (5.15)$$

Let  $\ell_{i+1}^{s_{a_r}}$  denote the length of the successful packet transmission subsection due to the user  $a_r$  in  $(i+1)$ th transmission epoch.  $\ell_{i+1}^s$  is the sum of subsections, i.e.,

$$\ell_{i+1}^s = \ell_{i+1}^{s_{a_1}} + \ell_{i+1}^{s_{a_2}} + \dots + \ell_{i+1}^{s_{a_n}}$$

$\pi_{j,n}^s(k_1)$  can therefore be written as

$$\begin{aligned} \pi_{j,n}^s(k_1) &= Prob[\ell_{i+1}^s = k_1 \mid \ell_i = j, \eta_{i+1} = n] = \\ &\sum Prob[\ell_{i+1}^{s_{a_1}} = k'_1, \ell_{i+1}^{s_{a_2}} = k'_2, \dots, \ell_{i+1}^{s_{a_n}} = k'_n \mid \ell_i = j, \eta_{i+1} = n] \end{aligned} \quad (5.16)$$

The above summation is over all possible values of  $k'_1, k'_2, \dots, k'_n$  such that their sum is equal to  $k_1$  and each of them is greater than or equal to one

But by definition  $\ell_{i+1}^{s_{a_r}} = b_{i+1}^{a_r} - 1 + 1 = b_{i+1}^{a_r}$ . One is subtracted here because the HOL packet is already counted in the collision resolution subsection and the second one is added to count for the last empty slot. Since each  $\ell_{i+1}^{s_{a_r}}$  is an independent random variable we have for  $n \leq k_1 \leq nj$

$$\begin{aligned} \pi_{j,n}^s(k_1) &= \sum_K Prob[b_{i+1}^{a_1} = k'_1, b_{i+1}^{a_2} = k'_2, \dots, b_{i+1}^{a_n} = k'_n \mid \ell_i = j, \eta_{i+1} = n] \\ &= \sum_K \xi(k'_1) \xi(k'_2) \dots \xi(k'_n) \end{aligned} \quad (5.17)$$

Substituting for each of  $\xi(k'_r)$  from Equation 5.15 into this equation we have

$$\begin{aligned} \pi_{j,n}^s(k_1) &= \frac{\sigma^{k_1} (1 - \sigma)^{jn - k_1}}{\{1 - (1 - \sigma)^j\}^n} \sum_K \binom{j}{k'_1} \binom{j}{k'_2} \dots \binom{j}{k'_n} \\ &= \frac{\sigma^{k_1} (1 - \sigma)^{jn - k_1}}{\{1 - (1 - \sigma)^j\}^n} C_{n,j}(k_1) \quad \text{for } n \leq k_1 \leq nj \end{aligned} \quad (5.18)$$

This establishes the theorem □

The following lemma of combinatorics enables us to compute the value of  $C_{n,j}(k_1)$  in a convenient way

**Lemma 5.2** Let  $K = \{(k'_1, k'_2, \dots, k'_n) \mid \sum_{r=1}^n k'_r = k_1, \forall k'_r \geq 1\}$  then

$$\sum_K \binom{j}{k'_1} \binom{j}{k'_2} \dots \binom{j}{k'_n} = \sum_{r=0}^{n-1} (-1)^r \binom{n}{r} \binom{(n-r)j}{k_1} \quad (5.19)$$

*Proof* The proof is straightforward and omitted here □

### 5.3.2 Determination of $\pi_n^c(k_2)$

$\pi_n^c(k_2)$  is the probability distribution of the collision resolution part of the epoch, when the number of active users at the beginning of the epoch is  $n$ . Within an epoch, the collision resolution algorithm employed by the HOL packet is the optimal collision resolution algorithm of Chapter 3.

Let us consider only the collision resolution part of the epoch ignoring the successful packet transmission part of the epoch. If  $n$  is the number of active users at the beginning of the epoch, then the initial system state for the collision resolution algorithm is  $(N, n)$ . Let  $p_{(N,n),(0,0)}(\Phi^*)$  denote the state transition probability from  $(N, n)$  to  $(0, 0)$  when the optimal control policy  $\Phi^*$  is employed. Let  $p_{(N,n),(0,0)}^{(k_2)}(\Phi^*)$  denote the corresponding  $k_2$ -step state transition probability, i.e.,

$$p_{(N,n),(0,0)}^{(k_2)}(\Phi^*) = \text{Prob}[S(k_2) = (0, 0) \mid S(0) = (N, n), \Phi^*] \quad (5.20)$$

Here  $S(0)$  is the initial state and  $S(k_2)$  is the state at the  $k_2$ th instant.

By definition,  $\pi_n^c(k_2)$  is the probability that the system reaches, for the first time, the state  $(0, 0)$  from the initial state  $(N, n)$  in  $k_2$  slots, i.e.,

$$\begin{aligned} \pi_n^c(k_2) = & \text{Prob}[S(k_2) = (0, 0), S(k_2 - 1) \neq (0, 0), \dots, S(1) \neq (0, 0) \mid S(0) = (N, n), \Phi^*] \end{aligned} \quad (5.21)$$

Clearly for  $k_2 = 1$  we have

$$\pi_n^c(1) = p_{(N,n),(0,0)}(\Phi^*) \quad (5.22)$$

Since the state  $(0, 0)$  is the absorbing state of the Markov chain characterizing the collision resolution algorithm, we have for  $k_2 > 1$

$$\begin{aligned} p_{(N,n),(0,0)}^{(k_2)}(\Phi^*) &= \sum_{r=1}^{k_2} \pi_n^c(r) p_{(0,0),(0,0)}^{(k_2-r)}(\Phi^*) \\ &= \sum_{r=1}^{k_2} \pi_n^c(r) \end{aligned} \quad (5.23)$$

Similarly we have

$$p_{(N,n),(0,0)}^{(k_2-1)}(\Phi^*) = \sum_{r=1}^{k_2-1} \pi_n^c(r) \quad (5.24)$$

Subtracting Equation 5.24 from Equation 5.23 we obtain

$$\pi_n^c(k_2) = p_{(N,n),(0,0)}^{(k_2)}(\Phi^*) - p_{(N,n),(0,0)}^{(k_2-1)}(\Phi^*) \quad (5.25)$$

We have thus proved the following lemma

**Lemma 5.3** *The probability distribution of the collision resolution length,  $\pi_n^c(k_2)$ , is given by*

$$\pi_n^c(k_2) = \begin{cases} p_{(N,n),(0,0)}^{(k_2)}(\Phi^*) & \text{for } k_2 = 1 \\ p_{(N,n),(0,0)}^{(k_2)}(\Phi^*) - p_{(N,n),(0,0)}^{(k_2-1)}(\Phi^*) & \text{for } k_2 > 1 \end{cases} \quad (5.26)$$

By substituting the values of  $\pi_n^c(k - k_1)$  and  $\pi_{j,n}^s(k_1)$  from Equations 5.26 and 5.10 in Equation 5.7, the state transition probabilities  $\pi_{j,k}$  can be evaluated. The state transition probability matrix  $\Pi = [\pi_{j,k}]$  can thus be constructed. Since the state space of the Markov chain  $\{\ell_i, i = 0, 1, \dots\}$  is infinite, this probability matrix  $\Pi$  is also of infinite order. This Markov chain  $\{\ell_i, i = 0, 1, \dots\}$  satisfies the following property

**Proposition 5.1** *The Markov chain  $\{\ell_i, i = 0, 1, \dots\}$  is irreducible and aperiodic*

*Explanation* A Markov chain is irreducible if all states communicate, i.e., every state can be reached from every other state. Let  $\pi_{j,k}^{(n)}$  denote the  $n$  step transition probability, then the state  $k$  can be reached from state  $j$  if and only if  $\pi_{j,k}^{(n)} > 0$  for some  $n \geq 1$ . Thus the Markov chain is irreducible if  $\pi_{j,k}^{(n)} > 0$  for some  $n \geq 1 \forall j, k \in \mathcal{Z}$ . By using the above expressions of the state transition probabilities, it can be easily verified that the Markov chain  $\{\ell_i, i = 0, 1, \dots\}$  is irreducible.

An irreducible Markov chain is aperiodic, if  $\pi_{j,j} > 0$  for some state  $j$ . This holds for the Markov chain under consideration. Thus  $\{\ell_i, i = 0, 1, \dots\}$  is irreducible and aperiodic.

## 5.4 Ergodicity of the Markov Chain

In this section, we find the conditions for the ergodicity of the Markov chain  $\{\ell_i, i = 0, 1, \dots\}$ . We first state the following Lemma due to Pakes [54] which will be used to establish the ergodicity

**Lemma 5.4** *Let  $\{X_n, n = 0, 1, \dots\}$  be an irreducible aperiodic Markov chain having as state space, the nonnegative integers. Let  $d_j$  be the drift at state  $j$  which is defined as*

$$d_j = E[X_{i+1} - X_i | X_i = j] \quad (5.27)$$

*Then for all  $j$  if  $|d_j| < \infty$  and if  $\limsup_{j \rightarrow \infty} d_j < 0$ , then  $\{X_n, n = 0, 1, \dots\}$  is ergodic*

**Theorem 5.2** *The transmission epoch length Markov chain  $\{\ell_i, i = 0, 1, \dots\}$  is ergodic for  $\sigma < 1/N$*

*Proof* We know from Section 5.3 that the chain  $\{\ell_i, i = 0, 1, \dots\}$  is irreducible and aperiodic. The drift at state  $j$  is given by

$$\begin{aligned} d_j &= E[\ell_{i+1} - \ell_i | \ell_i = j] \\ &= \sum_{n=0}^N \left\{ L^c(N, n) + \frac{jn\sigma}{1 - (1 - \sigma)^j} + \Delta_{0,n} \right\} p(n | j) - j \end{aligned} \quad (5.28)$$

The last equation can be written by making use of the fact that  $L^c(N, n) = \sum_{k_2=1}^{\infty} k_2 \pi_n^c(k_2)$

Hence we have

$$\begin{aligned} d_j &= \frac{j\sigma}{1 - (1 - \sigma)^j} \sum_{n=1}^N np(n | j) + \sum_{n=1}^N L^c(N, n)p(n | j) + p(0 | j) - j \\ &= j\sigma N + \sum_{n=1}^N L^c(N, n)p(n | j) + (1 - \sigma)^{jN} - j \\ &= j(\sigma N - 1) + (1 - \sigma)^{jN} + \sum_{n=1}^N L^c(N, n)p(n | j) \end{aligned} \quad (5.29)$$

Since  $N$  is the total number of unresolved users at the beginning of the epoch, we have

$$\begin{aligned} L^c(N, 1) &= 1 & L^c(N, N) &= N \\ \text{and } L^c(N, n) &< N & \text{for } 2 \leq n \leq N-1 \end{aligned}$$

Hence from Equation 5.29 we have

$$\begin{aligned} d_j &< N + (1 - \sigma)^j N + j(\sigma N - 1) \\ \Rightarrow |d_j| &< |N| + |(1 - \sigma)^j N| + |j(\sigma N - 1)| \\ &< \infty \quad \forall j \end{aligned} \tag{5.30}$$

This satisfies the first condition of the Pakes lemma. Now we have to show that  $\limsup_{j \rightarrow \infty} d_j < 0$ . Let

$$M_j = N + (1 - \sigma)^j N - j(1 - \sigma N) \tag{5.31}$$

Then we have

$$\begin{aligned} d_j &< M_j \\ \text{and also } \sup\{d_j, d_{j+1}, \dots\} &= M_j \end{aligned}$$

If  $\sigma N < 1$ , then from Equation 5.31  $\lim_{j \rightarrow \infty} M_j = -\infty$  and hence we have

$$\limsup_{j \rightarrow \infty} d_j = \lim_{j \rightarrow \infty} M_j = -\infty < 0 \tag{5.32}$$

Hence the Markov chain  $\{\ell_i, i = 0, 1, \dots\}$  is ergodic for  $\sigma N < 1$  or  $\sigma < 1/N$ . This establishes the result.  $\square$

**Remark 5.4.1** The above result is independent of any collision resolution algorithm as long as the expected collision resolution length  $L^c(N, n)$  is bounded. In our case if the total number of unresolved users at the beginning is  $N$ , then  $L^c(N, n) \leq N$ . Similar result has been obtained in [76] in the context of a different multiple access protocol.

## 5.5 Steady State Probabilities and Expected Transmission Epoch Length

### 5.5.1 Procedure for Computations

Let  $\pi_k$ s denote the steady state probabilities of the transmission epoch lengths, i.e.,

$$\pi_k = \lim_{i \rightarrow \infty} \pi_k^{(i)} = \lim_{i \rightarrow \infty} \text{Prob}[\ell_i = k] = \text{Prob}[\ell_\infty = k]$$

Since the Markov chain is ergodic, the steady state probabilities  $\pi_k$ s can be calculated by solving the following system of linear equations

$$\begin{aligned} \pi_k &= \sum_{j=1}^{\infty} \pi_j \pi_{j,k} \\ \text{and} \quad \sum_{k=1}^{\infty} \pi_k &= 1 \end{aligned} \quad (5.33)$$

Let  $\pi = [\pi_1, \pi_2, \dots]$  be the steady state probability vector and  $\Pi$  be the state transition probability matrix, then the above equation can be written in the matrix form as

$$\begin{aligned} \pi &= \pi \Pi \\ \sum_{k=1}^{\infty} \pi_k &= 1 \end{aligned} \quad (5.34)$$

The state transition matrix can be evaluated with the help of Equation 5.7 for given values of  $\sigma$  and  $N$ . Since the matrix  $\Pi$  and the steady state probability vector  $\pi$  are of infinite dimension, we truncate the probability vector and the state transition probability matrix to some finite dimension  $T$  to evaluate the steady state probability vector numerically. The appropriate value of  $T$  depends upon  $\sigma$  and  $N$  and is to be chosen carefully so that the effect of truncation error is small.

While truncating the state transition probability matrix  $\Pi = [\pi_{j,k}]$  to some finite dimension  $T$ ,  $\pi_{j,T}$  for  $j$  varying from 1 to  $T$  is set equal to  $(1 - \sum_{k=1}^{T-1} \pi_{j,k})$ . The truncated system of equations  $\pi_T = \pi_T \Pi$  and  $\sum_{k=1}^T \pi_k = 1$  can then be solved to give the probability vector  $\pi_T = [\pi_1, \pi_2, \dots, \pi_T]$ .

To solve this truncated system of equations, we use the iterative procedure [84, 53]. It is well known [46] that the iteration

$$\pi_T^{m+1} = \pi_T^m \Pi \quad (5.35)$$

will converge to the steady state probability vector  $\pi_T$  independent of the starting value  $\pi_T^0$ . In numerical computations, the iteration is stopped when the following error criterion is satisfied

$$|\pi_T^{n+1} - \pi_T^n| \leq \epsilon$$

where  $\epsilon$  is the specified tolerable error

After calculating the steady state probabilities, the expected transmission epoch length and the second moment of the epoch length under steady state conditions are given by

$$L_\infty = E[\ell_\infty] = \sum_{k=1}^{\infty} k \pi_k \quad (5.36)$$

$$L_\infty^2 = E[\ell_\infty^2] = \sum_{k=1}^{\infty} k^2 \pi_k \quad (5.37)$$

### 5.5.2 Numerical Results

For the numerical calculations, we assume the number of users  $N$  to be equal to 8. For  $N = 8$ , the system is stable for arrival probability  $\sigma$  less than  $1/N = 0.125$ . We calculate the steady state probability vector and the expected epoch length for various values of arrival probability  $\sigma$  from 0.01 to 0.1 for the truncated system. The choice of the truncation size is different for each arrival probability and is selected such that the expected transmission epoch length (and expected delay computed in Section 5.6) calculated by the truncated system remain unaltered to at least three decimal places, even when larger truncation size is used. Hence for each  $\sigma$ , we find out the appropriate truncation size. The values of truncation size for different arrival probabilities is given in Table 5.1. The size of 40 is considered sufficient for arrival probabilities less than 0.05. As the value of the arrival probability increases, the required truncation size also increases. The expected transmission epoch lengths and

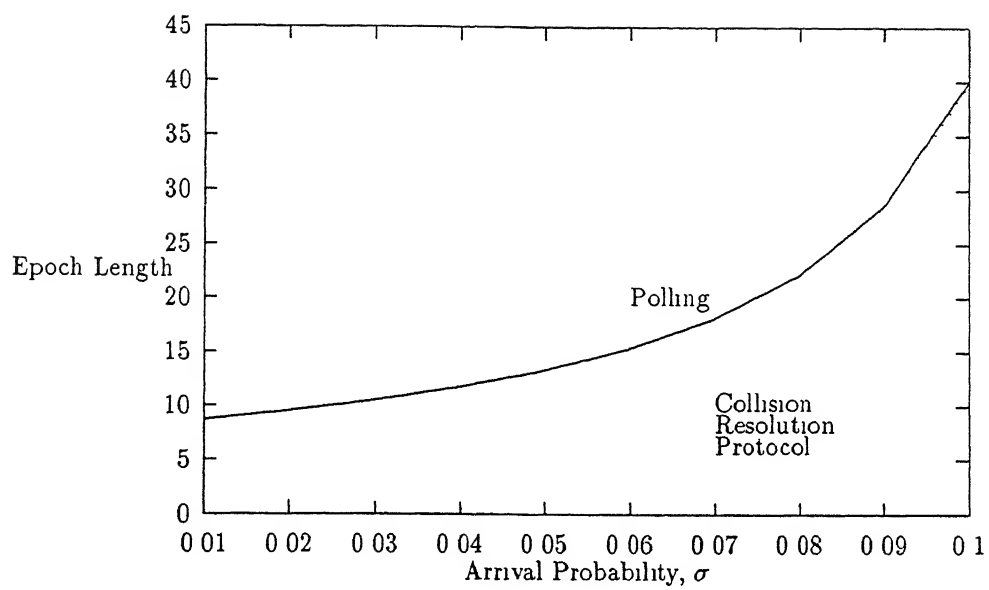


Figure 5.1 Expected Epoch Lengths

the second moments of the epoch length calculated by the truncated system are given in Table 5.2. For the arrival probability  $\sigma$  greater than 0.04, the expected epoch length increases very rapidly

Table 5.1 Truncation Size Chosen versus Arrival Probability

Arrival Probability $\sigma$	Truncation Size
0.01	40
0.02	40
0.03	40
0.04	40
0.05	50
0.06	50
0.07	50
0.08	60
0.09	80
0.10	100

Table 5.2 Expected Value and Variance of Transmission Epoch Length

Arrival Probability $\sigma$	Expected Transmission Epoch Length $L_{\infty}$	Second Moment of Epoch Length $L_{\infty}^2$	Variance of Epoch Length
0.01	1.0930	1.3154	0.1207
0.02	1.2259	1.8892	0.3863
0.03	1.4402	3.1046	1.0304
0.04	1.8519	6.3005	2.8709
0.05	2.8503	17.2353	9.1110
0.06	5.7564	62.0114	28.8752
0.07	12.5083	206.4339	49.9763
0.08	20.3183	452.4637	39.6303
0.09	27.9662	826.0169	43.9085
0.10	39.8011	1666.0550	81.9254

We know that the transmission epoch length consists of the collision reso-

lution part and the successful packet transmission part, i.e., we have

$$\ell_{i+1} = \begin{cases} \ell_{i+1}^c + \ell_{i+1}^s, & \text{for } \eta_{i+1} \geq 1 \\ 1 & \text{for } \eta_{i+1} = 0 \end{cases} \quad (5.38)$$

The corresponding expected values are given by

$$E[\ell_\infty | \eta_\infty = n] = L_\infty^c(N, n) + L_\infty^s(N, n) \quad \text{for } n \geq 1 \quad (5.39)$$

$$E[\ell_\infty | \eta_\infty = 0] = 1 \quad (5.40)$$

Thus  $E[\ell_\infty]$  can be written as

$$\begin{aligned} E[\ell_\infty] &= \sum_{n=1}^N (L_\infty^c(N, n) + L_\infty^s(N, n)) p(n) + p(0) \\ &= L_\infty^c + L_\infty^s + p(0) \end{aligned} \quad (5.41)$$

where  $p(n) = \text{Prob}(\eta_\infty = n)$  is the probability that the number of active users at the beginning of the transmission epoch under steady state condition is  $n$  and it is given by

$$p(\eta_\infty = n) = \sum_{j=1}^{\infty} p(n | j) \pi_j \quad (5.42)$$

The value of  $L_\infty^c$  can be calculated as

$$L_\infty^c = \sum_{j=1}^{\infty} \sum_{n=1}^N L^c(N, n) p(n | j) \pi_j \quad (5.43)$$

We note that  $L_\infty^s$  is the total number of packets transmitted successfully in a transmission epoch. Thus the ratio  $L_\infty^s/L_\infty$  gives the number of packets transmitted successfully per slot under the steady state operation or the *throughput* of the buffered system under the operation of our protocol. Similarly  $L_\infty^c$  is the number of slots "wasted" in collision resolution in a transmission epoch. The values of  $L_\infty^s$  and  $L_\infty^c$  are given in Table 5.3. The ratio  $L_\infty^c/L_\infty$  is a measure of the collision resolution efficiency of the protocol. We call it the *resolution factor* of the protocol. Note that <sup>the</sup> more the value of this resolution factor, <sup>the</sup> greater ~~will~~ be the slots used in collision resolution in a transmission epoch.

Now consider a TDMA system. In TDMA, there is a frame of  $N$  slots and each slot corresponds to a user. The user can transmit only one packet per

frame. If  $\sigma$  is the arrival probability, then the throughput of the TDMA system is  $N\sigma/N = \sigma$ . Similarly, the average number of slots wasted in a TDMA frame can be shown to be equal to  $N(1 - \sigma)$  and thus  $N(1 - \sigma)/N$  can be considered as the resolution factor of the TDMA system. Table 5.4 gives these values for different arrival probabilities for both the systems.

Table 5.3 Expected Length of Collision Resolution & Successful Packet Transmission Part

Arrival Probability $\sigma$	Expected Length of Collision Resolution Part $L_{\infty}^c$	Expected Length of Successful Packet Transmission Part $L_{\infty}^s$
0.01	0.0893	0.0875
0.02	0.2059	0.1962
0.03	0.3741	0.3458
0.04	0.6582	0.5933
0.05	1.2535	1.1448
0.06	2.7118	2.7849
0.07	5.3773	7.0570
0.08	7.2596	13.0520
0.09	7.8117	20.1544
0.10	7.9571	39.8440

**Remark 5.5.1** The problem of approximating the steady state probability vector of an infinite Markov chain by the steady state probability vector of a finite Markov chain arises in wide variety of problems. Some authors [15] have studied general methods for solving the steady state balance equations. Heyman [29] has derived sufficient conditions which imply that as the truncation size  $n$  tends to infinity, the stationary distribution of the truncated chains converge to the stationary distribution of the given chain. In the context of multiple access protocol, Nakasis and Ephremides [53] has encountered this problem while calculating the queue size distribution of buffered ALOHA. By utilizing the concept of dominant chain and by exploiting the particular nature of the problem, they have given a method to approximate an infinite Markov chain.

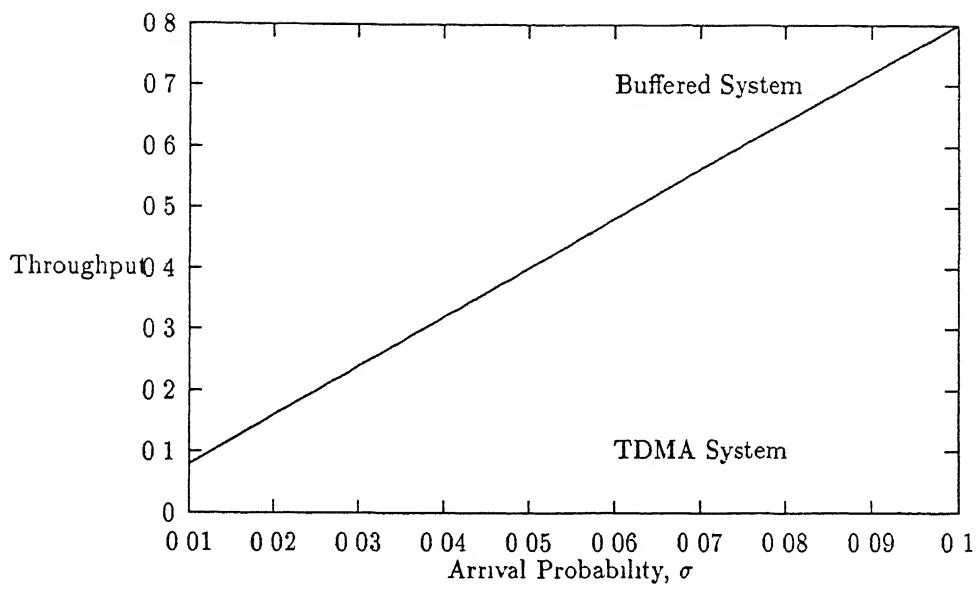


Figure 5.2 Throughput for Buffered and TDMA Systems

Table 5 4 Resolution Factor &amp; Throughput

Arrival Probability $\sigma$	Throughput of Buffered System $L_{\infty}^s/L_{\infty}$	Resolution Factor of Buffered System $L_{\infty}^c/L_{\infty}$	Throughput of TDMA System	Resolution Factor of TDMA System
0 01	0 080	0 076	0 01	0 99
0 02	0 160	0 167	0 02	0 98
0 03	0 240	0 259	0 03	0 97
0 04	0 320	0 355	0 04	0 96
0 05	0 401	0 439	0 05	0 95
0 06	0 483	0 471	0 06	0 94
0 07	0 564	0 429	0 07	0 93
0 08	0 642	0 357	0 08	0 92
0 09	0 720	0 279	0 09	0 91
0 10	0 800	0 199	0 10	0 90

In our numerical examples, we have used a technique similar to the one used by Wieselthier and Ephremides [84]. In our analysis, the steady state probabilities are finally needed to calculate the expected transmission epoch length and the expected packet delay. Both have an accuracy of at-least three decimal places for all arrival probabilities considered here. This is considered sufficient for our purposes.

## 5.6 Packet Delay Analysis

As explained in Chapter 2, the delay encountered by a packet is defined as the time from the moment of its generation to the moment of its successful transmission. Note that in the operation of our protocol, the packets generated in one transmission epoch are transmitted successfully in the next epoch. To calculate the average packet delay, we assume the steady state operation of the protocol.

Let us choose a packet randomly in an epoch and tag it. Suppose that our tagged packet has arrived in  $i$ th epoch and transmitted in  $(i + 1)$ th epoch. The packet arrival at any user in a slot is assumed to take place at the beginning

of the slot. Let  $n$  be the number of active users at the beginning of  $(i + 1)$ th epoch. Then  $i$ th epoch is called the *arriving epoch* and  $(i + 1)$ th epoch is called the *departing epoch*. The active user which contains the tagged packet is referred as the tagged user.

Let  $\delta$  be the delay of the tagged packet as shown in Figure 5.3. Then

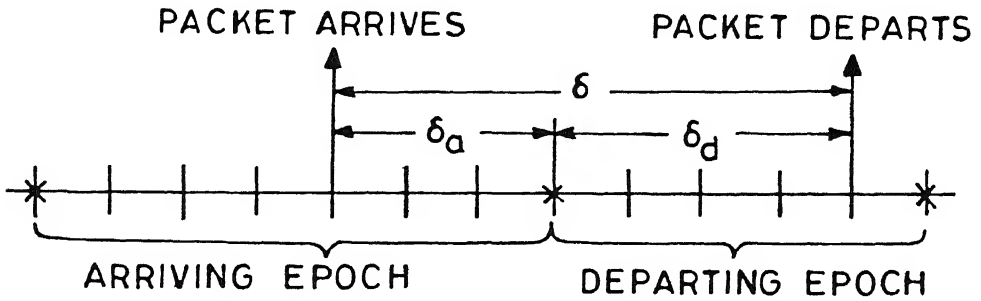


Figure 5.3 Delay encountered by a Packet

$$\delta = \delta^a + \delta^d \quad (5.44)$$

where  $\delta^a$  and  $\delta^d$  are the delays encountered by the tagged packet in the arriving epoch and the departing epoch respectively. Both  $\delta^a$  and  $\delta^d$  are random variables. The expected packet delay is

$$\begin{aligned} E[\delta] &= E[\delta^a] + E[\delta^d] \\ \mathcal{D} &= \mathcal{D}^a + \mathcal{D}^d \end{aligned} \quad (5.45)$$

where  $\mathcal{D}$ ,  $\mathcal{D}^a$  and  $\mathcal{D}^d$  are the expected values of  $\delta$ ,  $\delta^a$  and  $\delta^d$  respectively.

Let  $\delta_o^d$  and  $\delta_s^d$ , the components of the delay in the departing epoch, be defined as follows

$\delta_o^d$  = The delay due to the packets of other users which transmit before the tagged user plus the delay due to the collision resolution subsection of the tagged user

$\delta_s^d$  = The delay due to those packets of the tagged user which are ahead of the tagged packet

Since the delay due to HOL packet of the tagged user is already accounted in  $\delta_o^d$ , we subtract one slot unit time from  $\delta_s^d$ .  $\delta^d$  is then given by

$$\delta^d = \delta_o^d + (\delta_s^d - 1) + 1 \quad (5.46)$$

The last one is added to account for the tagged packet transmission time which is one slot unit. The corresponding expected values can be written as

$$\begin{aligned} E[\delta^d] &= E[\delta_o^d] + E[\delta_s^d] \\ \mathcal{D}^d &= \mathcal{D}_o^d + \mathcal{D}_s^d \end{aligned} \quad (5.47)$$

The delay  $\mathcal{D}$  is therefore given by

$$\mathcal{D} = \mathcal{D}^a + \mathcal{D}_s^d + \mathcal{D}_o^d \quad (5.48)$$

### 5.6.1 The Delay Components $\mathcal{D}^a$ and $\mathcal{D}_s^d$

Before we calculate the delay  $\mathcal{D}^a$  and  $\mathcal{D}_s^d$ , we state the following lemma

**Lemma 5.5** *Let  $p(\ell_a = j)$  be the probability that the length  $\ell_a$  of the arriving epoch is  $j$ . Then  $p(\ell_a = j)$  is given by*

$$p(\ell_a = j) = \frac{j \text{Prob}(\ell_\infty = j)}{E[\ell_\infty]} = \frac{j\pi_j}{L_\infty} \quad (5.49)$$

*Proof* The proof of this lemma follows from renewal theory. Under the steady state operation, starting instants of the transmission epochs form renewal instants. Kleinrock [37] has given the relationship between the probability of the selected interval in terms of the probability of a typical interval. The interval selected here is the arriving epoch. The typical epoch length under steady state is represented by  $\ell_\infty$ . Thus following [37], we can establish the above relationship  $\square$

The following theorems enable us to evaluate  $\mathcal{D}^a$  and  $\mathcal{D}_s^d$

**Theorem 5.3** *The expected delay  $\mathcal{D}^a$  is given by*

$$\mathcal{D}^a = \frac{E[\ell_\infty^2] + E[\ell_\infty]}{2E[\ell_\infty]} \quad (5.50)$$

where  $E[\ell_\infty]$  and  $E[\ell_\infty^2]$  are the first and second moments of the transmission epoch length under steady state operation and are given by Equation 5.36 and Equation 5.37

*Proof* Let  $\mathcal{D}^a(j)$  be the expected delay in the arriving epoch when the length of the epoch is  $j$ . Then  $\mathcal{D}^a$  can be expressed as

$$\mathcal{D}^a = \sum_{j=1}^{\infty} \mathcal{D}^a(j) p(\ell_a = j) \quad (5.51)$$

The delay of the tagged packet in the arriving epoch of length  $j$  can vary from 1 slot to  $j$  slots depending upon its arrival. As proved in Appendix B, the probability that the tagged packet has delay of  $k$  slots is  $1/j$  for all values of  $k$  from 1 to  $j$ . The expected delay  $\mathcal{D}^a(j)$  is therefore given by

$$\begin{aligned} \mathcal{D}^a(j) &= E[\delta^a \mid \ell_a = j] \\ &= \frac{1}{j} \sum_{k=1}^j k \\ &= \frac{j+1}{2} \end{aligned} \quad (5.52)$$

Substituting for  $\mathcal{D}^a(j)$  from Equation 5.52 and  $p(\ell_a = j)$  from Equation 5.49 into Equation 5.51, we get

$$\begin{aligned} \mathcal{D}^a &= \sum_{j=1}^{\infty} \frac{j(j+1)\pi_j}{2E[\ell_\infty]} \\ &= \frac{E[\ell_\infty^2] + E[\ell_\infty]}{2E[\ell_\infty]} \end{aligned} \quad (5.53)$$

This result can also be proved by considering that the delay in the arriving epoch is equivalent to the residual life in an renewal epoch

**Theorem 5.4** *The expected delay  $\mathcal{D}_s^d$  is given by*

$$\mathcal{D}_s^d = \frac{\sigma\{E[\ell_\infty^2] - E[\ell_\infty]\}}{2E[\ell_\infty]} \quad (5.54)$$

*Proof* The delay  $\mathcal{D}_s^d$  is the expected delay due to other packets of the tagged user which are ahead of the tagged packet. Let  $\mathcal{D}_s^d(j)$  be this expected delay conditioned on the length of the arriving epoch being equal to  $j$ . Then similar to Equation 5.52, we can write for  $\mathcal{D}_s^d$  as

$$\mathcal{D}_s^d = \sum_{j=1}^{\infty} \mathcal{D}_s^d(j) p(\ell_a = j) \quad (5.55)$$

The delay  $\mathcal{D}_s^d(j)$  is equal to the expected number of packets generated before the tagged packet when the length of the arriving epoch is  $j$ . Since a user generates a packet with probability  $\sigma$  in each slot, the expected number of packets generated before the tagged packet is  $\sigma$  times the expected number of slots occurring before the tagged packet in the epoch. If the length of the arriving epoch is  $j$  then the number of slots occurring before the tagged packet varies from 0 to  $j - 1$  with equal probability of  $1/j$ . Hence we have

$$\begin{aligned} \mathcal{D}_s^d &= E[\delta_s^d \mid \ell_a = j] \\ &= \sigma \frac{1}{j} \sum_{k=0}^{j-1} k \\ &= \frac{\sigma(j-1)}{2} \end{aligned} \quad (5.56)$$

Substituting Equation 5.56 and Equation 5.49 into Equation 5.55 we get

$$\begin{aligned} \mathcal{D}_s^d &= \frac{\sigma}{2} \sum_{j=1}^{\infty} \frac{j(j-1)\pi_j}{E[\ell_{\infty}]} \\ &= \frac{\sigma\{E[\ell_{\infty}^2] - E[\ell_{\infty}]\}}{2E[\ell_{\infty}]} \end{aligned} \quad (5.57)$$

This proves the theorem □

### 5.6.2 The Delay $\mathcal{D}_o^d$

The delay  $\mathcal{D}_o^d$  consists of two parts. One part is the expected delay due to the packets of other users which transmit before the tagged user. This part consists of the collision resolution subsection and the successful packet transmission subsection of each of these other users. The other part is due to the collision resolution subsection of the tagged user.

Like all transmission epochs, the departing epoch can be divided into  $n$  sections, if the number of active users at the beginning of the epoch is  $n$ . At the end of the first section, all unresolved users are reindexed as 1, 2, *etc.* in the order of their increasing address indices. It can be shown then that the departing epoch has several epochs nested into it. For example, if the departing epoch has state  $(N, n)$  at the beginning and state  $(M, n - 1)$  at the end of the first section, where  $M < N$ , then a new transmission epoch with initial state  $(M, n - 1)$  may be assumed to begin. Thus the departing epoch may be seen to consist of a transmission section and a new transmission epoch. Let

$$\mathcal{D}_o^d(j, N, n) = E[\delta_o^d \mid \ell_a = j, S(0) = (N, n)] \quad (5.58)$$

i.e.  $\mathcal{D}_o^d(j, N, n)$  is the expected value of  $\delta_o^d$  conditioned on the event that the length of the arriving epoch is  $j$  and the initial state of the system employing the collision resolution algorithm is  $(N, n)$ .  $\mathcal{D}_o^d$  is then given by

$$\mathcal{D}_o^d = \sum_{j=1}^{\infty} \sum_{n=0}^N \mathcal{D}_o^d(j, N, n) p(\eta_d = n, \ell_a = j) \quad (5.59)$$

where  $\eta_d$  is the number of active users at the beginning of the departing epoch and  $\ell_a$  is the length of the arriving epoch. From the above equation, we have

$$\mathcal{D}_o^d = \sum_{j=1}^{\infty} \sum_{n=0}^N \mathcal{D}_o^d(j, N, n) p(\eta_d = n \mid \ell_a = j) p(\ell_a = j) \quad (5.60)$$

Since  $\sigma$  is the arrival probability, we have

$$p(\eta_d = n \mid \ell_a = j) = \binom{N}{n} p^n (1 - p)^{N-n} \quad (5.61)$$

where  $p = 1 - (1 - \sigma)^j$

Here  $p(\ell_a = j)$  is the probability that the length of the epoch in which the tagged packet has arrived is  $j$  and can be calculated from Equation 5.49. In Equation 5.60, we have to calculate now  $\mathcal{D}_o^d(j, N, n)$

If we focus only on the collision resolution part of the epoch, after ignoring the successful packet transmission part, then the state reaching probability is defined to be the probability that the system ever reaches the state  $(M, n - 1)$

from the state  $(N, n)$ . Let this be denoted by  $\gamma\{(N, n), (M, n-1)\}$ . Let  $\beta\{(N, n), (M, n-1), \ell\}$  be the probability that the system reaches the state  $(M, n-1)$  from  $(N, n)$  in  $\ell$  slots. We then have

$$\gamma\{(N, n), (M, n-1)\} = \sum_{\ell=1}^{\infty} \beta\{(N, n), (M, n-1), \ell\} \quad (5.62)$$

In the above expression  $\{\beta\{(N, n), (M, n-1), \ell\}$  can be interpreted as the probability distribution of the collision resolution subsection of the first section.

We then have the following two lemmas

**Lemma 5.6** Let  $p_{(N,n),(M_1,n_1)}(\Phi^*)$  denote the state transition probability from the state  $(N, n)$  to  $(M_1, n_1)$  when optimal control policy  $\Phi^*$  is employed and  $\gamma\{(N, n), (M, n-1)\}$  be as defined above, then  $\gamma\{(N, n), (M, n-1)\}$  can be recursively computed by the following expression

$$\gamma\{(N, n), (M, n-1)\} = p_{(N,n),(M,n-1)}(\Phi^*) + \sum_{\mathcal{E}_M} p_{(N,n),(M_1,n_1)}(\Phi^*) \gamma\{(M_1, n_1), (M, n-1)\} \quad (5.63)$$

where  $\mathcal{E}_M = \{(M_1, n_1) \mid M_1 \neq M, n_1 \neq n-1\}$

*Proof* The proof is straightforward. From the state  $(N, n)$ , the Markov chain characterizing the collision resolution algorithm goes either to the state  $(M, n-1)$  and thus reaches the desired state or it goes to some other state  $(M_1, n_1)$  with the probability  $p_{(N,n),(M_1,n_1)}(\Phi^*)$  and from the state  $(M_1, n_1)$ , eventually the state  $(M, n-1)$  is reached with probability  $\gamma\{(M_1, n_1), (M, n-1)\}$ .  $\square$

**Lemma 5.7** Let  $L^{c_1}(N, n)$  be the expected length of the first collision resolution subsection of the epoch having state  $(N, n)$  at its beginning. Let  $L^c(N, n)$  be the expected length of the collision resolution part of the entire epoch having state  $(N, n)$  at its beginning, then  $L^{c_1}(N, n)$  is given by the following equation

$$L^{c_1}(N, n) = L^c(N, n) - \sum_{\mathcal{X}_M} L^c(M, n-1) \gamma\{(N, n), (M, n-1)\} \quad (5.64)$$

where  $\mathcal{X}_M = \{M \mid \beta\{(N, n), (M, n-1), \ell\} > 0\}$

*Proof* Let  $T_j$  for  $j = 1, 2, \dots, n$  denote the time instant where the  $j$ th section of an epoch ends and  $S(T_j)$  denote the state of the system employing the collision resolution algorithm at the instant  $T_j$ . Let  $L^c(N, n \mid S(T_1) = (M, n-1), \ell^c = \ell)$  denote the expected length of the collision resolution part of the epoch conditioned on the event that the first collision resolution section is of length  $\ell$  and the state at the end of the section is  $(M, n-1)$ . We can then write

$$L^c(N, n \mid S(T_1) = (M, n-1), \ell^c = \ell) = \ell + L^c(M, n-1) \quad (5.65)$$

By unconditioning we get

$$\begin{aligned} L^c(N, n) &= \sum_{\ell} \sum_{x_M} \ell \beta\{(N, n), (M, n-1), \ell\} \\ &\quad + \sum_{x_M} L^c(M, n-1) \gamma\{(N, n), (M, n-1)\} \\ &= L^c(N, n) + \sum_{x_M} L^c(M, n-1) \gamma\{(N, n), (M, n-1)\} \end{aligned} \quad (5.66)$$

Hence we have

$$L^c(N, n) = L^c(N, n) - \sum_{x_M} L^c(M, n-1) \gamma\{(N, n), (M, n-1)\} \quad (5.67)$$

□

**Definition 5.2** If any active user transmits successfully its HOL packet in the collision resolution subsection of any transmission section, it is said to be a winner of that section □

Let  $\alpha_1(N, n)$  be the probability that our tagged user is the *winner* in the first section of the epoch which has the state  $(N, n)$  at its beginning. Because of the nature of our collision resolution algorithm, any of the active users is equally likely to be the winner at the end of the collision resolution subsection of the first section. So the probability that the tagged active user is the winner in the first section of the epoch, when there were  $n$  active users at its beginning is just equal to  $1/n$ . Hence  $\alpha_1(N, n) = 1/n$ . We can now prove the following theorem which enables us to compute  $\mathcal{D}_o^d(j, N, n)$

**Theorem 5.5**  $\mathcal{D}_o^d(j, N, n)$  can be recursively computed by the following expression

$$\begin{aligned} \mathcal{D}_o^d(j, N, n) &= L^{c_1}(N, n) + (1 - \alpha_1(N, n)) \\ &\quad \left[ \frac{j\sigma}{1 - (1 - \sigma)^j} + \sum_{\mathcal{X}_M} \mathcal{D}_o^d(j, M, n - 1) \gamma\{(N, n), (M, n - 1)\} \right] \end{aligned} \quad (5.68)$$

where  $L^{c_1}(N, n)$  and  $\gamma\{(N, n), (M, n - 1)\}$  are as defined earlier. The set  $\mathcal{X}_M$  is the set of all  $M$  such that  $\gamma\{(N, n), (M, n - 1)\} > 0$ .

*Proof* By definition we have

$$\mathcal{D}_o^d(j, N, n) = E[\delta_o^d \mid \ell_a = j, \mathbf{S}(0) = (N, n)] \quad (5.69)$$

Therefore

$$\begin{aligned} \mathcal{D}_o^d(j, N, n) &= E[E[\delta_o^d \mid \ell_a = j, \mathbf{S}(0) = (N, n), \mathbf{S}(T_1) = (M, n - 1), \ell^{c_1} = \ell]] \\ &= E[\mathcal{D}_o^d(j, N, n \mid M, n - 1, \ell)] \end{aligned} \quad (5.70)$$

Here  $\mathcal{D}_o^d(j, N, n \mid M, n - 1, \ell)$  is the expected delay  $\mathcal{D}_o^d(j, N, n)$  conditioned on the event that state at the end of the first section is  $(M, n - 1)$  and the length of the first collision resolution section is  $\ell$ . For convenience, let this event be denoted by  $E_{M, \ell}$ .

The delay  $\delta_o^d$  is a random variable and its value conditioned on the event  $E_{M, \ell}$  can be written as

$$\delta_o^d(j, N, n \mid M, n - 1, \ell) = \ell \alpha_1(N, n) + (1 - \alpha_1(N, n)) (\ell + \ell^{s_1} + \delta_o^d(j, M, n - 1)) \quad (5.71)$$

This equation states that  $\delta_o^d$  is equal to the length of the first collision resolution subsection, if our tagged user is the winner of the first section which occurs with probability  $\alpha_1(N, n)$ . Otherwise it is equal to the length of the entire first section plus an additional delay  $\delta_o^d(j, M, n - 1)$ . After the first section, a new epoch may be assumed to begin with the initial state  $(M, n - 1)$  and the delay  $\delta_o^d(M, n - 1)$  is the delay in this new epoch which is nested into our departing epoch.

By taking expectation on both sides of Equation 5 71 we have

$$\mathcal{D}_o^d(j, N, n \mid M, n-1, \ell) = \ell + (1 - \alpha_1(N, n)) \left[ \frac{j\sigma}{1 - (1 - \sigma)^j} + \mathcal{D}_o^d(j, M, n-1) \right] \quad (5 72)$$

By unconditioning we get

$$\begin{aligned} \mathcal{D}_o^d(j, N, n) &= \sum_{\ell} \sum_{\mathcal{X}_M} \ell \beta\{(N, n), (M, n-1), \ell\} + (1 - \alpha_1(N, n)) \\ &\quad \left[ \frac{j\sigma}{1 - (1 - \sigma)^j} + \sum_{\ell} \sum_{\mathcal{X}_M} \beta\{(N, n), (M, n-1), \ell\} \mathcal{D}_o^d(j, M, n-1) \right] \\ &= L^c(N, n) + (1 - \alpha_1(N, n)) \\ &\quad \left[ \frac{j\sigma}{1 - (1 - \sigma)^j} + \sum_{\mathcal{X}_M} \mathcal{D}_o^d(j, M, n-1) \gamma\{(N, n), (M, n-1)\} \right] \quad (5 73) \end{aligned}$$

This recursive equation can be solved to yield  $\mathcal{D}_o^d(j, N, n)$  with the initial condition  $\mathcal{D}_o^d(j, M, 1) = 1 \forall M$   $\square$

### 5.6.3 Numerical Results

After calculating the value of  $\mathcal{D}_o^d(j, N, n)$  from Equation 5 68 and  $p(\ell_a = j)$  from Equation 5 49 and substituting these values in Equation 5 60,  $\mathcal{D}_o^d$  can be calculated. The expected packet delay is then the sum of  $\mathcal{D}_a$ ,  $\mathcal{D}_s^d$  and  $\mathcal{D}_o^d$ . The expected packet delay can thus be computed for various arrival rates for a given user population. The expected packet delays are given in Table 5 5 for  $N = 8$ .

We will compare these packet delays with the corresponding delays for specific polling type of protocol which is a variant of TDMA for buffered users. Before we perform the complete delay analysis of this protocol, the question before us is why we should choose this protocol for comparison. Tsybakov [75] has considered the broad philosophical framework under which different multiple access protocols can be compared. We take a brief look at this work and then proceed to the delay analysis of the polling protocol and its comparison with the protocol considered above for buffered users.

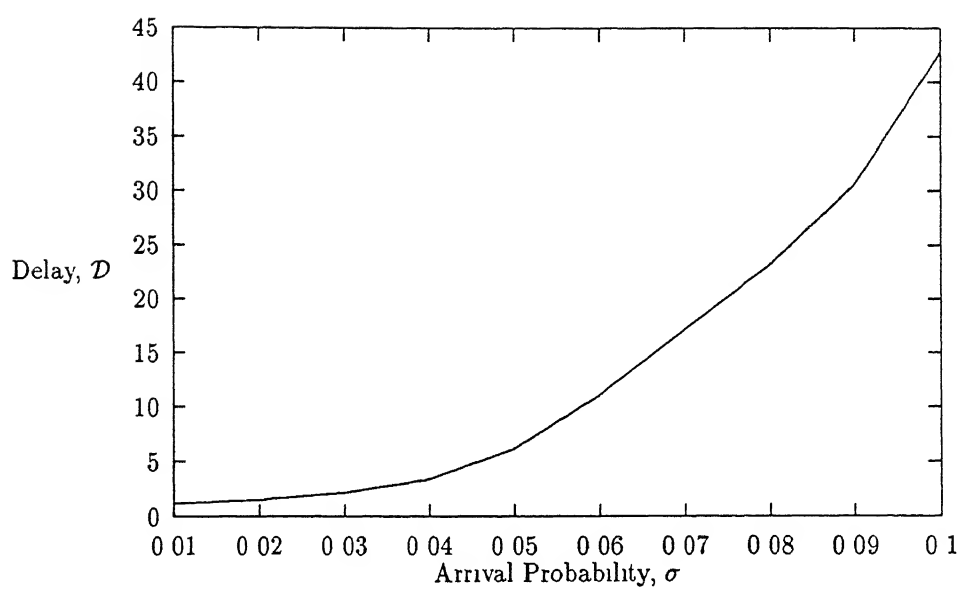


Figure 5.3 Packet Delays for Buffered Users

Table 5.5 Packet Delays for Buffered Users

Arrival Probability $\sigma$	Delay Component $\mathcal{D}_o^d$	Average Packet Delay $\mathcal{D}$
0.01	0.0986	1.2014
0.02	0.2610	1.5369
0.03	0.5636	2.1588
0.04	1.1917	3.4407
0.05	2.5440	6.1935
0.06	4.9808	11.1602
0.07	7.9170	17.2115
0.08	10.7286	23.2137
0.09	13.9961	30.5483
0.10	19.2844	42.7572

## 5.7 Polling Protocol

### 5.7.1 On Comparison of Multiple Access Protocols

As mentioned by Tsybakov [75], the multiple access protocols can be broadly divided into two types for comparison

- Station or User Control Protocols
- Packet Control Protocols

- 1 *User Control Protocols* Station or user control protocol establishes in what sequence and for how long the users may occupy the channel. According to this protocol, at any moment  $t$ , when a particular user occupying the channel releases the channel, the right to occupy the channel passes to another user whose address is determined by the protocol.

The efficiency of such user control protocols is low when a large proportion of time is wasted in passing the right to users that have nothing to transmit. Systems with user control protocols are various time sharing systems, polling systems.

- 2 *Packet Control Protocol* In packet control protocols, the protocol controls the transmission of the first packet queued in the buffer. The protocol determines the time instants when the transmission of the particular packet starts and ends. For a given packet of a user, these moments may depend on the previous starting and ending moments of the previous packets of that user. They may also depend sometimes upon the queue length information.

Although the collisions are unavoidable in such protocols, the efficiency of packet control protocols is generally high, when the users have bursty type of traffic. The protocols in this category try to gain information through collision slots and at the same time, the objective of the protocol is to allow successful transmission of packets.

Qualitatively speaking, generally, the efficiency of packet control protocols is high when the efficiency of user control protocols is low and vice-versa. Note that our access protocol described in this chapter belongs to the category of packet control protocol. We therefore consider a variant of TDMA for buffered users which is a user control protocol. The comparison of these two protocols lead to the conclusions about the relative performance of user control protocol over packet control protocol on a quantitative basis. The comparison criterion chosen is the expected packet delay.

### 5.7.2 Description of Polling Protocol

In this section, we discuss a polling protocol for finite buffered users. This protocol can be considered as a variant of TDMA. As before, we have a finite population of  $N$  users, each with a buffer of infinite size. The packet arrival process to each user is a discrete time Bernoulli process and the probability of a packet arrival in any slot at each user is  $\sigma$ . The users are addressed as  $1, 2, \dots, N$ .

As before, the operation of the protocol can be divided into successive transmission epochs. The packets that arrived in one epoch are transmitted in the next epoch. The protocol grants users the right to access the channel

in the order of their increasing address numbers. Thus user addressed as 1 transmits its packets first, user 2 next and so on till at the end, user  $N$  transmits. If a user has nothing to transmit, it leaves the slot allocated to it as empty and the protocol gives the transmission right to the next user. The user indicates the end of its packet transmissions by an empty slot in the end. Thus in every epoch, there are  $N$  empty slots each corresponding to a user and at the end of each empty slot, the transmission right passes to the next user.

Let  $Z_{i+1}$  be the length of  $(i+1)$ th transmission epoch of this protocol. This length is equal to  $N$  plus the total number of packets that arrived to the system in the previous epoch. Thus

$$Z_{i+1} = N + R_{i+1} \quad (5.74)$$

where  $R_{i+1}$  is the total number of packets present in the buffers of all the users at the beginning of  $(i+1)$ th epoch. Note that the epoch length is always greater than or equal to  $N$ . Since only the packets that arrived in the previous epoch are transmitted in the current epoch, the length of the current epoch depends upon the length of the previous epoch only. Thus the sequence  $\{Z_i, i = 0, 1, \dots\}$  is a Markov chain with state space  $\{N, N+1, \dots\}$ .

### 5.7.3 Evaluation of State Transition Probabilities

The state transition probabilities of the Markov chain  $\{Z_i, i = 0, 1, \dots\}$  can be expressed as follows.

Let  $q_{j,k}$  be the probability of transition from the state  $j$  to the state  $k$ , i.e.,

$$q_{j,k} = \text{Prob}[Z_{i+1} = k \mid Z_i = j] \quad (5.75)$$

Since  $Z_{i+1} = N + R_{i+1}$ , we have

$$\begin{aligned} q_{j,k} &= \text{Prob}[Z_{i+1} = k \mid Z_i = j] \\ &= \text{Prob}[R_{i+1} = k - N \mid Z_i = j] \end{aligned} \quad (5.76)$$

Assuming the packet arrival processes to different users to be independent of each other,  $\text{Prob}[R_{i+1} = k - N \mid Z_i = j]$  can be written as the  $N$ -fold

convolution of the probabilities of the number of packets in the buffers of the users. Let  $q_j^{(r)}(k_1)$  be the probability that the number of packets in the buffer of  $r$ th user at the beginning of the epoch is  $k_1$  given that the length of the previous epoch is  $j$ , i.e.,

$$\begin{aligned} q_j^{(r)}(k_1) &= \text{Prob}[b_{i+1}^r = k_1 \mid Z_i = j] \\ &= \binom{j}{k_1} \sigma^{k_1} (1 - \sigma)^{j-k_1} \end{aligned} \quad (5.77)$$

where  $b_{i+1}^r$  is the number of packets in the buffer of user  $r$  at the beginning of  $(i + 1)$ th epoch. But from Equation 5.76

$$\begin{aligned} q_{j,k} &= \text{Prob}[R_{i+1} = k - N \mid Z_i = j] \\ &= q_j^{(1)}(k - N) * q_j^{(2)}(k - N) * \dots * q_j^{(N)}(k - N) \\ &= \binom{Nj}{k - N} \sigma^{k - N} (1 - \sigma)^{Nj - k - N} \quad \text{for } N \leq k \leq N + Nj \end{aligned} \quad (5.78)$$

where  $*$  is the convolution operator.

The transition diagram for this Markov chain is shown in Figure 5.4. If the Markov chain is in the state  $j$  at some instant, then the next state will be  $(j + 1), (j + 2), \dots$  etc. depending upon whether one packet arrived or two packets arrived and so on, at the system. If the present state is  $j$ , the maximum number of packets that can arrive is  $Nj$ , hence the next state in that case will be  $N(j + 1)$ .

#### 5.7.4 Steady State Probability Vector

Arguing along the lines similar to that of Section 5.4, it can be proved that the Markov chain  $\{Z_i, i = 0, 1, \dots\}$  is ergodic. Because of the ergodic nature of the Markov chain, we have a unique steady state probability vector. Let  $q = [q_N, q_{N+1}, \dots]$  denote the steady state probability vector, where  $q_k$  is given by

$$q_k = \lim_{i \rightarrow \infty} \text{Prob}[Z_i = k], \quad (5.79)$$

where  $k = N, N + 1, \dots$

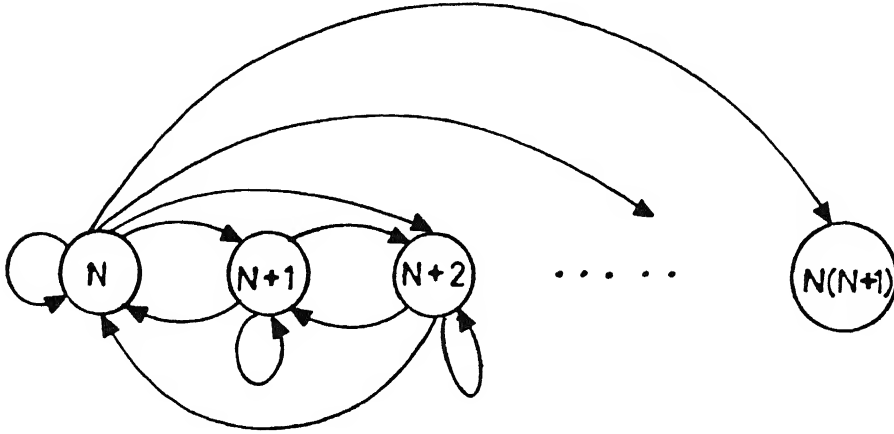


Figure 5.4 State Transition diagram for TDMA for buffered users

Note that  $q_0, q_1, \dots, q_{N-1}$  are zero. Let  $Q$  denote the state transition probability matrix of this Markov chain. The steady state probability vector can then be solved by the following balance equations

$$\mathbf{q} = \mathbf{q}Q \quad (5.80)$$

$$\text{and} \quad \sum_{k=N}^{\infty} q_k = 1 \quad (5.81)$$

The first and second moments of the epoch length are then given by

$$E[Z_{\infty}] = \sum_{k=N}^{\infty} k q_k \quad (5.82)$$

$$\text{and} \quad E[Z_{\infty}^2] = \sum_{k=N}^{\infty} k^2 q_k \quad (5.83)$$

### 5.7.5 Numerical Results

In this section, we perform the numerical calculations in a way similar to that of our earlier protocol for buffered users in Section 5.5.2. Thus we first calculate the steady state probability vector and the expected epoch length for different values of arrival probability for the truncated system. The truncation size is chosen in a similar manner. These sizes are listed in Table 5.6. The expected

length and the variance of the epoch length calculated by the truncated system are shown in Table 5.7. For comparison purposes we have shown the values of the expected epoch length from Table 5.2. Note that the epoch length for our TDMA type of polling protocol is always greater than the total number of users, *i.e.*, greater than 8 as these many number of empty slots will always be contained in any epoch for this protocol.

### 5.7.6 Expected Packet Delay

As defined previously, the expected packet delay is the expected value of the delay of a packet which is the time from the moment of its generation to the moment of its successful transmission. We proceed to calculate the expected delay along the lines of Section 5.6.

Thus as before, consider a tagged packet in the buffer of an active user at the beginning of an epoch. Let the address of this user containing the tagged packet be  $M$ . Unlike our previous access protocol, where the delay of a packet is independent of the user's address, in this polling protocol, the expected delay is user dependent.

The delay of the tagged packet consists of the following parts

- 1 The delay due to waiting in the arriving epoch, its expected value being denoted by  $\mathcal{D}_p^a$ .
- 2 The delay in the departing epoch due to the packets of other users which transmit before the user  $M$ , its expected value being denoted by  $\mathcal{D}_{p_o}^d(M)$ .
- 3 The delay in the departing epoch due to other packets of the user  $M$  which are ahead of the tagged packet. The expected value of this delay is denoted by  $\mathcal{D}_{p_s}^d$ .
- 4 The transmission time of the tagged packet which is one slot unit time.

Let  $\mathcal{D}_p(M)$  denote the total expected delay at the user  $M$ . Then we have

$$\mathcal{D}_p(M) = \mathcal{D}_p^a + \mathcal{D}_{p_o}^d + \mathcal{D}_{p_s}^d + 1 \quad (5.84)$$

Table 5.6 Truncation Size vs Arrival Probability

Arrival Probability $\sigma$	Truncation Size
0.01	40
0.02	40
0.03	40
0.04	40
0.05	50
0.06	50
0.07	60
0.08	80
0.09	100
0.10	100

Table 5.7 Expected Length &amp; Variance of Epoch Length (Also shown in this table are the Expected Transmission Epoch Length values for our protocol for Buffered users)

Arrival Probability $\sigma$	Expected Epoch Length for Polling $Z_{\infty}$	Variance of Epoch Length for Polling	Expected Transmission Epoch Length $L_{\infty}$
0.01	8.6956	0.6939	1.0930
0.02	9.5238	1.5328	1.2259
0.03	10.5263	2.6010	1.4402
0.04	11.7647	4.0280	1.8519
0.05	13.3333	6.0359	2.8503
0.06	15.3847	9.0244	5.7564
0.07	18.1820	13.8047	12.5083
0.08	22.2231	22.1879	20.3183
0.09	28.5733	38.9342	27.9662
0.10	40.0039	80.1229	39.8011

We can state the following propositions which enable us to calculate  $\mathcal{D}_p^a$  and  $\mathcal{D}_{ps}^d$ .

**Proposition 5.2** *The delay  $\mathcal{D}_p^a$  is given by*

$$\mathcal{D}_p^a = \frac{E[Z_\infty^2] + E[Z_\infty]}{2 E[Z_\infty]} \quad (5.85)$$

**Proposition 5.3** *The delay  $\mathcal{D}_{ps}^d$  is given by*

$$\mathcal{D}_{ps}^d = \frac{\sigma\{E[Z_\infty^2] - E[Z_\infty]\}}{2 E[Z_\infty]} \quad (5.86)$$

*Proofs* The proofs of these propositions can be carried out in a manner similar to that of Theorems 5.3 and 5.4. Note that these delays are independent of the user's address.  $\square$

Let the length of the epoch in which our tagged packet arrives be denoted by  $Z_a$ . Let  $\mathcal{D}_{po,j}^d(M)$  be the expected delay due to the packets of other users which transmit before the user  $M$  and conditioned on the length of the arriving epoch  $Z_a$  being equal to  $j$ . Then  $\mathcal{D}_{po}^d(M)$  can be written as

$$\mathcal{D}_{po}^d(M) = \sum_{j=N}^{\infty} \mathcal{D}_{po,j}^d(M) p(Z_a = j) \quad (5.87)$$

where  $p(Z_a = j)$  is the probability that the length of the arriving epoch is  $j$ . This probability similar to Equation 5.49 is given by

$$p(Z_a = j) = \frac{j q_j}{E[Z_\infty]} \quad (5.88)$$

Before the user  $M$ , a total of  $(M - 1)$  users will be given the rights to transmit. Corresponding to each user, we have only one empty slot. The average number of packets that have arrived in the previous epoch of length  $j$  in the buffer of these users and will be transmitted before the user  $M$  is  $j(M - 1)\sigma$ . Thus there is a total delay equal to the sum of  $M - 1$  and  $j(M - 1)\sigma$ . Hence we have

$$\begin{aligned} \mathcal{D}_{po,j}^d(M) &= (M - 1) + j(M - 1)\sigma \\ &= (M - 1)[1 + j\sigma] \end{aligned} \quad (5.89)$$

By substituting the expression for  $\mathcal{D}_{po,j}^d(M)$  in Equation 5.87,  $\mathcal{D}_{po}^d(M)$  can be calculated

### Numerical Calculations of Packet Delay

By calculating  $\mathcal{D}_p^a$ ,  $\mathcal{D}_{ps}^d$  and  $\mathcal{D}_{po}^d(M)$ , the expected packet delay at the user  $M$  can be calculated from Equation 5.84. Unlike in the case of the protocol described earlier in this chapter, the packet delay for TDMA protocol for buffered users depends upon the user's address or index. Note that the packet delay essentially has three components—out of which the two namely  $\mathcal{D}_p^a$  and  $\mathcal{D}_{ps}^d$ , are independent of the user's address while the third component  $\mathcal{D}_{po}^d(M)$  depends upon the user's address. Also note that the value of  $\mathcal{D}_{po}^d(M)$  is the delay due to other users which are ahead of the user  $M$ . The value of this delay component for the first user is zero. Thus the delay at the user 1 is simply the value of  $(\mathcal{D}_p^a + \mathcal{D}_{ps}^d + 1)$ . The delay  $\mathcal{D}_{po}^d(M)$  increases linearly with the user's address.

Table 5.8 gives the values of  $\mathcal{D}_p^a$ ,  $\mathcal{D}_{ps}^d$  and  $\mathcal{D}_{po}^d(M)$  for various values of arrival probabilities for  $M = 2$ . This table also shows the average packet delay at the user 2. The delay at the user 1 gives the best case delay for this protocol while the delay at the user 8 gives the worst case delay. We compare these delays with the expected delays for the protocol employing the collision resolution algorithm. These values are tabulated in Table 5.9. We note that below the arrival probability of 0.04, the delay due to the protocol employing the collision resolution algorithm (CRP) is better than that of the best case for the polling protocol.

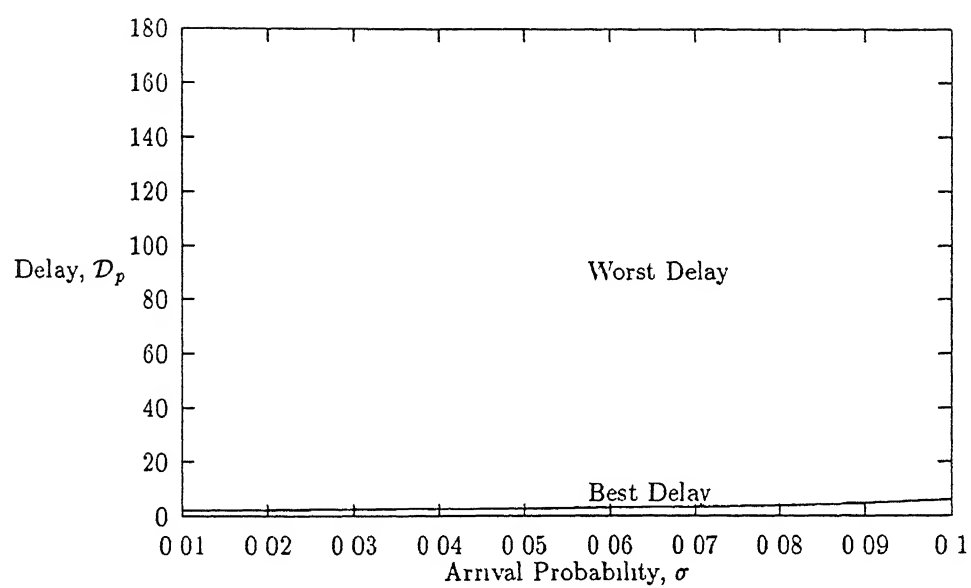


Figure 5.4 Comparison of Packet Delays

Table 5.8 Packet Delays due to Polling Protocol for Buffered Users

Arrival Probability $\sigma$	Delay $\mathcal{D}_p^a + \mathcal{D}_{ps}^d$	Delay $\mathcal{D}_{po}^d(M = 2)$	Average Packet Delay $\mathcal{D}_p(M)$
0.01	1.0877	4.9265	7.0143
0.02	1.1936	5.4292	7.6229
0.03	1.3232	6.0332	8.3564
0.04	1.4842	6.7756	9.2599
0.05	1.6892	7.7126	10.4018
0.06	1.9582	8.9347	11.8930
0.07	2.3258	10.5985	13.9244
0.08	2.8577	12.9996	16.8573
0.09	3.6942	16.7700	21.4642
0.10	5.2006	23.5537	29.7543

Table 5.9 Comparison of Packet Delays

Arrival Probability $\sigma$	Delay for CRP	Best Delay for Polling Protocol	Worst Delay for Polling Protocol
0.01	1.2014	2.0877	36.5732
0.02	1.5369	2.1936	40.1980
0.03	2.1588	2.3232	44.5556
0.04	3.4407	2.4842	49.9134
0.05	6.1935	2.6892	56.6774
0.06	11.1602	2.9582	65.5011
0.07	17.2115	3.3258	77.5153
0.08	23.2137	3.8577	94.8549
0.09	30.5483	4.6942	122.0842
0.10	42.7572	6.2006	171.0765

## Chapter 6

# An Optimal Collision Resolution Algorithm for Multipacket Reception

---

The previous chapters have discussed the random access algorithms for the conventional random access systems which can be modelled by the *Common Receiver Model*. In such systems, when two or more packets are transmitted in a slot, they are destroyed. In this chapter, we extend our discussion to random access systems with multipacket reception capability. For this, we consider the CDMA-RA system model introduced in Section 2.6.1. In CDMA-RA system, even if two or more packets are transmitted in a slot, some of them may be received successfully. The unsuccessful users may employ a collision resolution algorithm for retransmission. In this model, the packets are transmitted by encoding them with codes chosen from a set of codes with low cross-correlation properties. These are called address codes. As explained in Section 2.6.1, various schemes like transmitter based coding, receiver based coding and hybrid schemes are possible for assigning these addresses. Here we consider a kind of transmitter based code assignment. The CDMA-RA system can then be modelled by a Generalized Common Receiver Model.

In this chapter, we first discuss the assumptions made for the generalized common receiver model. As in Chapter 3, we focus on the blocked random access algorithms for this generalized model. Specifically, we address the question of determining an optimal collision resolution algorithm. We extend

the formulation of optimal collision resolution algorithm of Chapter 3 to the generalized common receiver model. Similar to that of Chapter 3, we define an appropriate state of the system and control variables that constitute a Markovian Decision Process and argue that the problem of optimal collision resolution algorithm is equivalent to the first passage problem of this Markovian Decision Process. The dynamic programming technique is then applied to derive an optimal algorithm which minimizes the expected collision resolution length. The algorithm is illustrated with numerical examples.

## 6.1 The System Model

We consider a finite user common receiver model of CDMA-RA system as shown in Figure 6.1. There are many sources of bursty data traffic which send

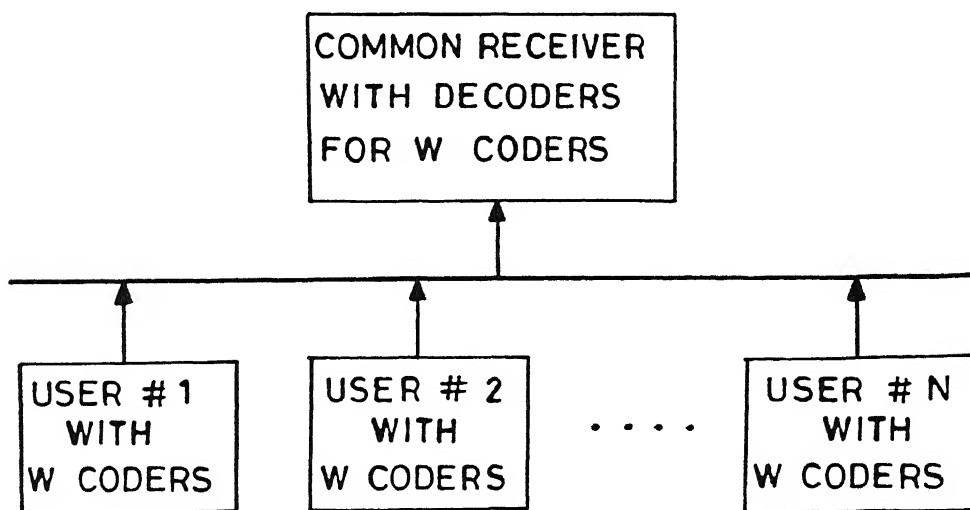


Figure 6.1 Generalized Common Receiver Model

fixed length packets to the receiver through a common broadcast communication channel. These packets are, however, encoded with address codes as explained below. The other assumptions about this model are

- 1 There are total  $N$  users in the system. They are addressed as  $1, 2, \dots, N$
- 2 The slotted operation of the system is assumed
- 3 *Address Codes* Each of the  $N$  users is equipped with a set of  $W$  orthogonal codes. The number of orthogonal codes  $W$  is much smaller than the total number of users in the system (i.e.,  $W \ll N$ ). Let this set of codes be denoted by  $\{1, 2, \dots, W\}$ . Each user before transmitting its packet, encodes it with one of the  $W$  orthogonal codes which is specified by the random access algorithm. In this chapter, we have illustrated the random access algorithm with  $W = 2$ . The case of  $W > 2$  can be easily generalized
- 4 The common receiver has the decoder for all the  $W$  codes. Thus the common receiver is in fact a bank of  $W$  receivers and each of them can receive packets transmitted using the respective codes
- 5 The transmission of two or more packets in a slot using the same code results in collision at the receiver. However, if only one packet is using a code in a slot, then that packet is received successfully. Thus at-most  $W$  packets can be transmitted successfully in a slot if all these  $W$  packets are using the distinct codes
- 6 *Feedback Vector* It is assumed that at the end of each slot, all users know about the state of the slot through feedback information. The feedback, however, is assumed to be a  $W$ -dimensional vector. Each element of this vector corresponds to a particular code and assumes ternary values of 0, 1 or 2 depending upon whether no packet was transmitted, or only one packet was transmitted or more than one packets were transmitted on the corresponding code in the slot
- 7 The feedback information is assumed to be received errorless by all users
- 8 *Blocked Access Operation* We assume that the users employ the blocked random access algorithm. As we know already that in blocked access, the operation of the system can be divided into successive transmission

intervals called epochs. In each epoch, the contending users resolve their collisions by using a collision resolution algorithm. The packets generated in one epoch are transmitted in the next epoch.

- 9 *Single Packet per Epoch* Each user has only two buffers. The first buffer contains the packet which is undergoing the collision resolution in the current epoch, while the second buffer stores any newly generated packet. All other packets which are generated after this new packet, are assumed to be lost. In other words, a user is assumed to generate and transmit only one packet per epoch.
- 10 *Active Users* As defined in Chapter 3, an active user is defined to be a user if it has a packet waiting in its buffer for transmission. We assume that the number of active users at the beginning of a collision resolution epoch is known to all users.

## 6.2 Random Access Algorithm for CDMA-RA System

We know from Chapter 3 that the collision resolution algorithm completely characterizes the blocked random access algorithm within the class  $\mathcal{A}_1^N$ . If the users transmit only one packet per epoch, then the collision resolution algorithm is given by the function  $f[i, \Theta(t)]$ . The function  $f[\cdot]$  assumes binary values of either 1 or 0 and it specifies whether the user  $i$  should transmit its packet at the instant  $t$  or not. In other words, the algorithm can be specified by an ensemble  $\{B(t)\}$  where  $B(t) = \{i \mid f[i, \Theta(t)] = 1\}$ .

In the same way, for the generalized common receiver model, the collision resolution algorithm can be defined by the function  $f[i, C_i, \Theta(t)]$  where  $i$  as before is the user's address and  $C_i$  is the code to be used by the user  $i$  for encoding its packet. The code  $C_i$  can be one of the  $W$  orthogonal codes. Thus the collision resolution algorithm for the CDMA-RA model can be specified by the ensemble  $\{B(t)\}$  where  $B(t) = [B_1(t), B_2(t), \dots, B_W(t)]$  and  $B_r(t) = \{i \mid f[i, C_i = r, \Theta(t)] = 1\}$ . Thus the collision resolution algorithm

specifies the set of users (for each address code) which are allowed to transmit their packets using the respective codes

Our interest in this chapter is in the determination of the optimal collision resolution algorithm for the generalized common receiver model. But before we show how to determine this algorithm, we first study the dynamics of our system. For this, we define the state of the system

### 6.2.1 The System State

We first generalize the definition of a resolved user in this section

**Definition 6.1** *A user is said to be a resolved one if one of the followings conditions is obtained*

- 1 *It is known to all other users that the user does not have a packet*
- 2 *The user was enabled in a slot with some code and this code was used by at-most one packet for transmission*

Suppose some users are enabled to transmit with the same code. If only one of these users has a packet, then the position in the feedback vector corresponding to this particular code would indicate a success. Thus, though the identity of the user having a packet does not become known to all other users, these users are treated as resolved. Similarly, the users enabled to transmit with some different code may get resolved. The total number of resolved users would be equal to the sum of the users resolved on each code.

As in Chapter 3, the state of the system at the instant  $t$  is defined to be a two tuple  $S(t) = [N(t), n(t)]$  where as before,  $N(t)$  is the total number of unresolved users and  $n(t)$  is the total number of active users at the instant  $t$ . If the number of active users at the beginning of a collision resolution epoch is  $n$ , then since all users are unresolved at the beginning, we have the state  $S(0) = (N, n)$

### 6.2.2 Dynamic Evolution of the System State

The collision resolution algorithm for the CDMA-RA system can be considered as a control policy. This control policy comprises a sequence of control actions to be taken at each stage. However, for this generalized system, the policy specifies the number of users to be enabled separately for each code.

Let  $\{S(t), t = 0, 1, \dots\}$  denote the sequence of observed states. Let  $\{U(t), t = 0, 1, \dots\}$  denote the sequence of control actions taken. Since we have assumed that there are  $W$  orthogonal codes, the control action at any instant is a  $W$ -dimensional vector. Thus the control  $U(t)$  at the instant  $t$  is given by  $U(t) = [u_1(t), u_2(t), \dots, u_W(t)]$  where  $u_r(t)$  is the number of users enabled to transmit at the time  $t$  using the code  $r$ . Let  $\{\theta(t), t = 0, 1, \dots\}$  denote the sequence of observed feedbacks. The feedback  $\theta(t)$  is also a  $W$ -dimensional vector and can be written as  $\theta(t) = [\theta_1(t), \theta_2(t), \dots, \theta_W(t)]$  where each  $\theta_r(t) \forall 1 \leq r \leq W$  assumes ternary values of 0, 1 or 2 depending upon whether no packet is transmitted, only one packet is transmitted or more than one packets are transmitted on the code  $r$ .

As in Chapter 3, let  $\mathcal{U}(S_i)$  denote the set of admissible control actions when the state of the system is  $S_i = (N_i, n_i)$ . Then the set of admissible control actions consist of all such  $W$  tuples  $(u_1, u_2, \dots, u_W)$  such that  $U = u_1 + u_2 + \dots + u_W \leq N_i$  and each  $u_r$  lies between 0 and  $N_i$ . In other words,

$$\mathcal{U}(S_i) = \left\{ (u_1, u_2, \dots, u_W) \sum_{r=1}^W u_r = U \leq N_i \text{ and } 0 \leq u_r \leq N_i, \forall 1 \leq r \leq W \right\}$$

We also use the notation  $\mathcal{U}(S(t))$  to denote the set of control actions at the instant  $t$ .

To understand the evolution of the system state, we assume  $W = 2$ . Then the control action is  $U(t) = [u_1(t), u_2(t)]$ . Let  $u_1(t) = u_1$ ,  $u_2(t) = u_2$  and  $U = u_1 + u_2$ . This value  $U$  is then the total number of users to be enabled. For this value, there will be many possible sets of users that can be enabled. In the optimal collision resolution algorithm of Chapter 3, the users choose one such set randomly by using a pseudorandom number generator. Thus similar to that of Chapter 3, the set of  $U$  users in which  $u_1$  select the code 1 and  $u_2$  select the code 2 is also chosen randomly.

By enabling these  $U$  users, a feedback  $\theta(t+1) = [\theta_1(t+1), \theta_2(t+1)]$  is observed. Since each of these elements  $\theta_i(t+1)$  can take ternary values,  $\theta(t+1)$  can take total of 9 values. If  $\theta_1(t+1) = \theta_2(t+1) = 0$ , it means the enabled  $u_1$  and  $u_2$  users do not have any packet and they are treated as resolved and so the next state moves to  $(N_i - (u_1 + u_2), n_i)$ . Similarly, if  $\theta_1(t+1) = 1$  and  $\theta_2(t+1) = 0$ , then it means that amongst the enabled  $u_1$  users, only one user had a packet, while among the  $u_2$  users, none had a packet so that  $u_1$  and  $u_2$  users are resolved and so the next state moves to  $(N_i - (u_1 + u_2), n_i - 1)$ . In this way, we can describe the evolution of the system state.

The dynamic evolution of the system can be summarized in the following steps

- The stochastic process consisting of the sequence of states  $\{S(t) = [N(t), n(t)], t = 0, 1, \dots\}$  is called the core process.
- If  $n$  is the number of active users at the beginning of the collision resolution epoch, then the initial state  $S(0) = (N, n)$  is known to all users.
- The stochastic process consisting of the sequence of observed feedbacks  $\{\theta(t), t = 0, 1, \dots\}$  is called the observation process. The feedback  $\theta(t)$  is multidimensional. For  $W = 2$ ,  $\theta(t) = [\theta_1(t), \theta_2(t)]$ .
- When the control  $U(t) = [u_1(t), u_2(t)]$  is applied at the time  $t$ , the state  $S(t)$  undergoes a transition to the state  $S(t+1)$  and feedback  $\theta(t+1) = [\theta_1(t), \theta_2(t)]$  is observed.
- The probability of transition to the next state  $S(t+1)$  depends only upon the present state  $S(t)$  and the control action  $U(t)$ . For a given sequence of control actions, the sequence  $\{S(t), t = 0, 1, \dots\}$  is a Markov chain. The state transition probabilities can be evaluated as explained later.
- If the state  $S(t)$  is known, then by observing the feedback  $\theta(t+1)$  and the control action  $U(t)$ , the state  $S(t+1)$  can be determined exactly. Since  $S(0)$  is assumed to be known to all users, it follows that all users have perfect knowledge of the state at every stage.

- The stochastic process  $\{S(t), U(t), t = 0, 1, \dots\}$  with the augmented control action  $U(t)$  is a Markovian Decision Process

### 6.3 State Transition Probabilities

In this section, we calculate the probability of transition from the state  $S_i = (N_i, n_i)$  to some other state  $S_j$ , when the control action is employed. For illustration purposes, we take  $W = 2$ . These probabilities can be easily calculated for values of  $W$  greater than 2 also. However, for higher values of  $W$ , the computation becomes tedious due to the size of the admissible control values.

Let  $p_{S_i, S_j}(u_1, u_2)$  denote the probability of transition from the state  $S_i = (N_i, n_i)$  to some state  $S_j = (N_j, n_j)$ , when the control applied is  $[u_1, u_2]$ . We consider the following cases

- 1 For  $n_i = 2$ . When  $n_i = 2$  and the control applied is  $[u_1, u_2]$  then the next state can be  $(u_1, 2)$  or  $(u_2, 2)$  or  $(N_i - u_1 - u_2, 1)$  or  $(N_i - u_1 - u_2, 2)$  or  $(0, 0)$  (The probabilities of transitions to remaining states are zero). These probabilities can be expressed as

$$p_{(N_i, 2), (u_1, 2)}(u_1, u_2) = \frac{\binom{u_1}{2}}{\binom{N_i}{2}} \quad (6.1)$$

$$p_{(N_i, 2), (u_2, 2)}(u_1, u_2) = \frac{\binom{u_2}{2}}{\binom{N_i}{2}} \quad (6.2)$$

$$p_{(N_i, 2), (N_i - u_1 - u_2, 2)}(u_1, u_2) = \frac{\binom{N_i - u_1 - u_2}{2}}{\binom{N_i}{2}} \quad (6.3)$$

$$p_{(N_i, 2), (N_i - u_1 - u_2, 1)}(u_1, u_2) = \frac{u_1}{\binom{N_i}{2}} + \frac{u_2}{\binom{N_i}{2}} \quad (6.4)$$

$$p_{(N_i, 2), (0, 0)}(u_1, u_2) = \frac{u_1 u_2}{\binom{N_i}{2}} \quad (6.5)$$

If  $N_i - u_1 - u_2$  is equal to  $u_1$  or  $u_2$  then the probability of transition to

$(u_1, 2)$  or  $(u_2, 2)$  will be given by

$$p_{(N_i, 2), (u_1, 2)}(u_1, u_2) = 2 \frac{\binom{u_1}{2}}{\binom{N_i}{2}} \quad \text{for } N_i - u_1 - u_2 = u_1 \quad (6.6)$$

$$p_{(N_i, 2), (u_2, 2)}(u_1, u_2) = 2 \frac{\binom{u_2}{2}}{\binom{N_i}{2}} \quad \text{for } N_i - u_1 - u_2 = u_2 \quad (6.7)$$

$$(6.8)$$

The probabilities of transitions from  $(N_i, 2)$  to all other states will be zero. This fact can be expressed as

$$p_{(N_i, 2), (N_j, n_j)}(u_1, u_2) = 0 \quad (6.9)$$

for  $N_j \neq u_1$  or  $u_2$  or  $N_i - u_1 - u_2$  and  $n_j \neq 2$  or 1 or 0

- 2 For  $n_i = 3$  When  $n_i = 3$  and the control  $[u_1, u_2]$  is applied then the next states can be  $(N_i - u_1, 3)$  or  $(N_i - u_2, 3)$  or  $(N_i - u_1 - u_2, 2)$  or  $(u_2, 2)$  or  $(u_1, 2)$  or  $(N_i - u_1 - u_2, 1)$ . The probability of transition to these states can be written as

$$p_{(N_i, 3), (N_i - u_1, 3)}(u_1, u_2) = \frac{\binom{N_i - u_1}{3}}{\binom{N_i}{3}} \quad (6.10)$$

$$p_{(N_i, 3), (N_i - u_2, 3)}(u_1, u_2) = \frac{\binom{N_i - u_2}{3}}{\binom{N_i}{3}} \quad (6.11)$$

$$p_{(N_i, 3), (N_i - u_1 - u_2, 2)}(u_1, u_2) = \frac{(N_i - u_1 - u_2)(u_1 + u_2)}{\binom{N_i}{3}} \quad (6.12)$$

$$p_{(N_i, 3), (u_1, 2)}(u_1, u_2) = \frac{\binom{u_1}{2} u_2}{\binom{N_i}{3}} \quad (6.13)$$

$$p_{(N_i, 3), (u_2, 2)}(u_1, u_2) = \frac{\binom{u_2}{2} u_1}{\binom{N_i}{3}} \quad (6.14)$$

$$p_{(N_i, 3), (N_i - u_1 - u_2, 1)}(u_1, u_2) = \frac{u_1}{\binom{N_i}{3}} + \frac{u_2}{\binom{N_i}{3}} \quad (6.15)$$

The probabilities of transitions from  $(N_i, 3)$  to all other states will be zero

3 For  $n_i = 4$  For this, the state transition probabilities can be written as

$$p_{(N_i, 4), (v_1 + u_2, 4)}(u_1, u_2) = \frac{\binom{u_1}{2} \binom{u_2}{2}}{\binom{N_i}{4}} \quad (6.16)$$

$$p_{(N_i, 4), (N_i - u_1 - u_2, 4)}(u_1, u_2) = \frac{\binom{N_i - u_1 - u_2}{4}}{\binom{N_i}{4}} \quad (6.17)$$

$$p_{(N_i, 4), (N_i - u_1, 4)}(u_1, u_2) = \frac{\sum_{r_2=2}^{u_2} \binom{u_2}{r_2} \binom{N_i - u_1 - u_2}{4 - r_2}}{\binom{N_i}{4}} \quad (6.18)$$

$$p_{(N_i, 4), (N_i - u_2, 4)}(u_1, u_2) = \frac{\sum_{r_1=2}^{u_1} \binom{u_1}{r_1} \binom{N_i - u_1 - u_2}{4 - r_1}}{\binom{N_i}{4}} \quad (6.19)$$

$$p_{(N_i, 4), (N_i - u_1 - u_2, 2)}(u_1, u_2) = \frac{u_1 u_2 \binom{N_i - u_1 - u_2}{2}}{\binom{N_i}{4}} \quad (6.20)$$

$$p_{(N_i, 4), (N_i - u_1 - u_2, 3)}(u_1, u_2) = \frac{u_1 \binom{N_i - u_1 - u_2}{3}}{\binom{N_i}{4}} + \frac{u_2 \binom{N_i - u_1 - u_2}{3}}{\binom{N_i}{4}} \quad (6.21)$$

$$p_{(N_i, 4), (N_i - u_1, 3)}(u_1, u_2) = \frac{\sum_{r_2=2}^{u_2} u_1 \binom{u_2}{r_2} \binom{N_i - u_1 - u_2}{3 - r_2}}{\binom{N_i}{4}} \quad (6.22)$$

$$p_{(N_i, 4), (N_i - u_2, 3)}(u_1, u_2) = \frac{\sum_{r_1=2}^{u_1} u_2 \binom{u_1}{r_1} \binom{N_i - u_1 - u_2}{3 - r_1}}{\binom{N_i}{4}} \quad (6.23)$$

$$p_{(N_i, 4), (N_i - u_1 - u_2, 2)}(u_1, u_2) = \frac{u_1 u_2 \binom{N_i - u_1 - u_2}{2}}{\binom{N_i}{4}} \quad (6.24)$$

4 For other cases The state transition probabilities from  $(N_i, n_i)$  for other values of  $n_i$  will be given by the following expressions

$$p_{(N_i, n_i), (N_i - u_1 - u_2, n_i)}(u_1, u_2) = \frac{\binom{N_i - u_1 - u_2}{n_i}}{\binom{N_i}{n_i}} \quad (6.25)$$

$$p_{(N_i, n_i), (N_i - u_1, n_i)}(u_1, u_2) = \frac{\sum_{r_2=2}^{u_2} \binom{u_2}{r_2} \binom{N_i - u_1 - u_2}{n_i - r_2}}{\binom{N_i}{n_i}} \quad (6.26)$$

$$p_{(N_i, n_i), (N_i - u_2, n_i)}(u_1, u_2) = \frac{\sum_{r_1=2}^{u_1} \binom{u_1}{r_1} \binom{N_i - u_1 - u_2}{n_i - r_1}}{\binom{N_i}{n_i}} \quad (6.27)$$

$$p_{(N_i, n_i), (N_i - u_1 - u_2, n_i - 2)}(u_1, u_2) = \frac{u_1 u_2 \binom{N_i - u_1 - u_2}{n_i - 2}}{\binom{N_i}{n_i}} \quad (6.28)$$

$$p_{(N_i, n_i), (N_i - u_1 - u_2, n_i - 1)}(u_1, u_2) = \frac{(u_1 + u_2) \binom{N_i - u_1 - u_2}{n_i - 1}}{\binom{N_i}{n_i}} \quad (6.29)$$

$$p_{(N_i, n_i), (N_i - u_1, n_i - 1)}(u_1, u_2) = \frac{\sum_{r_2=2}^{u_2} u_1 \binom{u_2}{r_2} \binom{N_i - u_1 - u_2}{n_i - 1 - r_2}}{\binom{N_i}{n_i}} \quad (6.30)$$

$$p_{(N_i, n_i), (N_i - u_2, n_i - 1)}(u_1, u_2) = \frac{\sum_{r_1=2}^{u_1} u_2 \binom{u_1}{r_1} \binom{N_i - u_1 - u_2}{n_i - 1 - r_1}}{\binom{N_i}{n_i}} \quad (6.31)$$

$$p_{(N_i, n_i), (N_i, n_i)}(u_1, u_2) = \frac{\sum_{r_1=2}^{u_1} \sum_{r_2=2}^{u_2} \binom{u_1}{r_1} \binom{u_2}{r_2} \binom{N_i - u_1 - u_2}{n_i - r_1 - r_2}}{\binom{N_i}{n_i}} \quad (6.32)$$

## 6.4 Determination of Optimal Collision Resolution Algorithm

As in Chapter 3, the collision resolution algorithm is a control policy and the optimal algorithm minimizes the expected time to reach the target state where all users are resolved. Suppose a policy  $\Phi(c)$  is used. Similar to that of Section 3.4.1, we introduce a cost structure into our model. Let  $\Psi[S_i, u_1, u_2]$  be the expected cost incurred when the state is  $S_i$  and the control action is  $[u_1, u_2]$ .

We define the cost  $\Psi$  as follows

$$\begin{aligned} \Psi[S_i, u_1, u_2] &= 1 \quad \forall S_i \neq (0, 0) \text{ and } u_1, u_2 \in \mathcal{U}(S_i) \\ \text{and } \Psi[(0, 0), u_1, u_2] &= 0 \end{aligned} \quad (6.33)$$

When the state  $(0, 0)$  is reached, then all users are resolved and the collision resolution epoch ends. If we assume that  $p_{(0,0),(0,0)}(\Phi(c)) = 1$ , then the total expected cost can be expressed as follows

$$\begin{aligned} C_{\Phi(c)}(S(0) = (N, n)) &= \\ &\sum_{t=0}^{\infty} \sum_{S_i \in \mathcal{S}} \sum_{u_1, u_2 \in \mathcal{U}(S_i)} \text{Prob}[S(t) = S_i, \mathbf{u}(t) = [u_1, u_2] \mid S(0) = (N, n)] \\ &\quad \Psi[S_i = (N_i, n_i), u_1, u_2] \end{aligned} \quad (6.34)$$

The cost  $C_{\Phi(c)}(S(0) = (N, n))$  is the expected collision resolution length. The optimal collision resolution algorithm minimizes this expected collision resolution length.

As before, let  $\mathcal{Y}(c)$  denote the class of possible control policies for our generalized random access model and  $\mathcal{Y}_D(c)$  be the subclass consisting of stationary deterministic policies. Then similar to Theorem 3.2, we have the following proposition.

**Proposition 6.1** *If the cost structure is defined as in Equation 6.33, then there exists an optimal collision resolution algorithm which minimizes the expected collision resolution length. This optimal control policy  $\Phi^*(c)$  is a stationary deterministic policy.*

The optimal control policy can be computed by the following proposition.

**Proposition 6.2** *Let  $V_0(S_i) = 0 \quad \forall S_i \in [\{\mathcal{S}\} - (0, 0)]$ . Define the following*

$$V_{m+1}(S_i) = \min_{u_1, u_2 \in \mathcal{U}(S_i)} \left[ \Psi[S_i, u_1, u_2] + \sum_{\substack{S_j \in \mathcal{S} \\ S_j \neq (0, 0)}} p_{S_i, S_j}(u_1, u_2) V_m(S_j) \right] \quad (6.35)$$

*then  $\lim_{m \rightarrow \infty} V_m(S_i) = C_{\Phi^*(c)}(S(0) = (N_i, n_i)) = L^c(N_i, n_i)$  where  $\Phi^*(c) \in \mathcal{Y}_D$  minimizes  $C_{\Phi(c)}(S(0) = (N, n))$ .*

*Proof.* The proof of these propositions can be carried out similar to those of Theorems 3.2 and 3.3.

### 6.4.1 Numerical Results

Equation 6.35 can be solved iteratively and the optimal control actions can be computed for each possible state. For the purpose of illustration, we consider  $N = 8$  and  $N = 16$ . The state transition probabilities can be calculated from Section 6.3. The iteration is performed using Equation 6.35 and is stopped when the following condition is satisfied

$$|V_{m+1}(S_i) - V_m(S_i)| < 1 \times 10^{-3} \quad \forall S_i \in \mathcal{S}$$

Table 6.1 Expected Collision Resolution Length and Optimal Control for  $N = 8$ 

State	Expected Collision Resolution Length	Optimal Control	
		$u_1$	$u_2$
(8, 8)	4 000	1	1
(8, 7)	4 000	1	1
(8, 6)	3 857	1	1
(8, 5)	3 569	2	2
(8, 4)	2 977	2	2
(8, 3)	2 407	2	2
(8, 2)	1 571	4	4
(8, 1)	1 000	1	7
(7, 7)	4 000	1	1
(7, 6)	3 714	1	1
(7, 5)	3 400	1	2
(7, 4)	2 937	1	2
(7, 3)	2 348	2	2
(7, 2)	1 571	3	4
(7, 1)	1 000	1	6
(6, 6)	3 000	1	1
(6, 5)	3 000	1	1
(6, 4)	2 733	1	1
(6, 3)	2 350	2	3
(6, 2)	1 533	3	3
(6, 1)	1 000	1	5
(5, 5)	3 000	1	1
(5, 4)	2 600	1	1
(5, 3)	2 200	1	2
(5, 2)	1 500	2	3
(5, 1)	1 000	1	4
(4, 4)	2 000	1	1
(4, 3)	2 000	1	1
(4, 2)	1 333	2	2
(4, 1)	1 000	1	3
(3, 3)	2 000	1	1
(3, 2)	1 333	1	2
(3, 1)	1 000	1	2
(2, 2)	1 000	1	1
(2, 1)	1 000	1	1
(1, 1)	1 000	0	1

Table 6.2 Expected Collision Resolution Length and Optimal Control for  $N = 16$ 

State	Expected Collision Resolution Length	Optimal Control	
		$u_1$	$u_2$
(16,1)	1 000	1	15
(16,2)	1 733	8	8
(16,3)	2 643	6	6
(16,4)	3 476	4	4
(16,5)	4 279	3	3
(16,6)	5 024	3	3
(16,7)	5 661	2	2
(16,8)	6 251	2	2
(16,9)	6 809	2	2
(16,10)	7 293	2	2
(16,11)	7 605	2	2
(16,12)	7 786	2	2
(16,13)	7 899	2	2
(16,14)	7 966	1	1
(16,15)	8 000	1	1
(16,16)	8 000	1	1
(15,1)	1 000	1	14
(15,2)	1 733	7	8
(15,3)	2 631	5	6
(15,4)	3 438	4	4
(15,5)	4 237	3	3
(15,6)	4 957	2	3
(15,7)	5 587	2	2
(15,8)	6 164	2	2
(15,9)	6 676	1	2
(15,10)	7 059	1	2
(15,11)	7 331	1	2
(15,12)	7 540	1	2
(15,13)	7 715	1	2
(15,14)	7 866	1	1
(15,15)	8 000	1	1

*Continued-*

State	Expected Collision Resolution Length	Optimal Control	
		$u_1$	$u_2$
(14,1)	1 000	1	13
(14,2)	1 725	7	7
(14,3)	2 605	5	5
(14,4)	3 410	3	3
(14,5)	4 164	3	3
(14,6)	4 843	2	2
(14,7)	5 437	2	2
(14,8)	5 986	2	2
(14,9)	6 449	2	2
(14,10)	6 713	2	2
(14,11)	6 867	2	2
(14,12)	6 956	1	1
(14,13)	7 000	1	1
(14,14)	7 000	1	1
(13,1)	1 000	1	12
(13,2)	1 717	6	7
(13,3)	2 575	4	5
(13,4)	3 364	3	3
(13,5)	4 115	2	3
(13,6)	4 768	2	2
(13,7)	5 352	2	2
(13,8)	5 843	1	2
(13,9)	6 204	1	2
(13,10)	6 462	1	2
(13,11)	6 670	1	2
(13,12)	6 846	1	1
(13,13)	7 000	1	1

*Continued*

State	Expected Collision Resolution Length	Optimal Control	
		$u_1$	$u_2$
(12,1)	1 000	1	11
(12,2)	1 697	6	6
(12,3)	2 572	4	5
(12,4)	3 332	3	3
(12,5)	4 019	2	2
(12,6)	4 614	2	2
(12,7)	5 171	2	2
(12,8)	5 590	2	2
(12,9)	5 817	2	2
(12,10)	5 939	1	1
(12,11)	6 000	1	1
(12,12)	6 000	1	1
(11,1)	1 000	1	10
(11,2)	1 690	5	6
(11,3)	2 511	4	4
(11,4)	3 254	3	3
(11,5)	3 953	2	2
(11,6)	4 544	2	2
(11,7)	5 018	1	2
(11,8)	5 354	1	2
(11,9)	5 609	1	2
(11,10)	5 818	1	1
(11,11)	6 000	1	1
(10,1)	1 000	1	9
(10,2)	1 666	5	5
(10,3)	2 475	3	3
(10,4)	3 185	3	3
(10,5)	3 834	2	2
(10,6)	4 360	2	2
(10,7)	4 732	2	2
(10,8)	4 911	1	1
(10,9)	5 000	1	1
(10,10)	5 000	1	1

*Continued*

State	Expected Collision Resolution Length	Optimal Control	
		$u_1$	$u_2$
(9,1)	1 000	1	8
(9,2)	1 638	4	5
(9,3)	2 463	2	3
(9,4)	3 127	2	3
(9,5)	3 756	2	2
(9,6)	4 193	1	2
(9,7)	4 523	1	2
(9,8)	4 777	1	1
(9,9)	5 000	1	1

Table 6.3 Expected Collision Resolution Length for  $N = 16$ 

State	Expected Collision Resolution Length
(16,1)	1 000
(16,2)	1 733
(16,3)	2 643
(16,4)	3 476
(16,5)	4 279
(16,6)	5 024
(16,7)	5 661
(16,8)	6 251
(16,9)	6 809
(16,10)	7 293
(16,11)	7 605
(16,12)	7 786
(16,13)	7 899
(16,14)	7 966
(16,15)	8 000
(16,16)	8 000

# Chapter 7

## Conclusions

---

With the widespread use of computer communication networks, there has been a considerable interest in the field of random access communication and several random access algorithms have been proposed. However, most of these studies are not directed towards developing a theory of random access algorithms. This may be attributed, in part, to the ever increasing pressure of network designer community to come up with practically useful protocols.

A major contribution in random access communication has been made by Tsybakov [78] who looked at the structure of the information pattern available to the users and defined a class of random access algorithms by giving a formal notion of a random access algorithm for the infinite user model. This thesis started with the motivation of investigating, in a similar manner, random access algorithms for the finite user model. The focus of the thesis is limited to only a sub-class of these algorithms and one of the important conclusions is that it is possible to derive an optimal random access algorithm over this sub-class under a set of not quite unrealistic assumptions. We summarize below some of the findings of the thesis to conclude our discussions.

Tsybakov's definition of random access algorithm is the starting point of this investigation. We have given a unified review of random access algorithms within Tsybakov's formulation. We then define a random access algorithm for the finite user model. It is shown how different random access algorithms can be classified using this definition. It is not clear at present what kind of relationships exist between the class of algorithms for the finite user and

infinite user model, though as remarked in Chapter 2 of the thesis, it is possible to give a general definition of a random access algorithm applicable to both finite and infinite user model

The determination of an optimal algorithm over the entire class of random access algorithms is, in general, a very difficult problem. We have therefore restricted our attention to only a subclass of these algorithms. Our interest in this class is motivated by the appealing conjecture of Tsybakov that an optimal algorithm in this class is also optimal over the entire class for the infinite user model. Another interesting aspect of the algorithms over this class is that they can be shown to consist of a channel access algorithm and a collision resolution algorithm. We have considered only those random access algorithms which employ the blocked channel access. In other words, we have frozen the channel access scheme to be the blocked access *a priori* and thus the operation of the system can be divided into successive collision resolution epochs. Furthermore, we have assumed that users can transmit only one packet per epoch. We then systematically investigate collision resolution algorithm. Two main results are borne out of this investigation

- Firstly, when the number of active users at the beginning of a collision resolution epoch is known, then it is shown that an appropriate set of state and control variables, that constitute a Markovian Decision Process (MDP), can be identified and the problem of optimal collision resolution algorithm is equivalent to the first passage problem of this MDP. It is shown that there exists an optimal algorithm which is a stationary and deterministic control policy that can be derived using dynamic programming technique. We have illustrated the algorithm with numerical examples and presented the steady state analysis of this algorithm.
- Secondly, when the number of active users at the beginning is not known exactly but only probabilistically, then it is shown that the probability distribution of the state conditioned on the past information constitute the appropriate new state description. Thus the finite user random access system is shown to have an alternative representation where the optimal collision resolution algorithm can be formulated as the first

passage problem with this new state. Due to the computational burden in the implementation of this algorithm, we have not pursued this course further but suggested two suboptimal algorithms. These algorithms are based upon updating the conditional probability vector with the help of control action and observed feedback at each stage. One of the algorithms (SOA1) chooses control action corresponding to the state which has the maximum value of probability while the other algorithm (SOA2) chooses control action corresponding to the expected state. The performance of these algorithms is studied by simulations with different initial probability distributions. These probabilities are chosen to represent a range of users from lightly loaded to heavily loaded. One of the important observations of these simulations is that the degradation in performance for the two suboptimal algorithms with respect to a bound on the optimal performance is not significant. If the optimal algorithm, when the initial state is known only probabilistically, were known then it would have performed worse than this bound. It is therefore reasonable to conclude that these suboptimal algorithms are quite satisfactory.

We next considered the case of users with buffers. The operation of the system can again be divided into successive transmission intervals called transmission epochs. The head of the line packet (HOL) of each user participates in the collision resolution using the collision resolution algorithm mentioned above. As soon as any of these HOL packets is transmitted successfully, the algorithm is interrupted and those packets of the user which were generated in the previous epoch are transmitted in consecutive slots. The algorithm then starts again from the point where it is interrupted. A detailed steady state analysis of this protocol is carried out. It is shown that under certain conditions, the sequence of the transmission epoch lengths is an ergodic Markov chain. The steady state probabilities and the first and second moments of the transmission epoch length are numerically evaluated. The packet delay analysis is then performed. The expected packet delay consists of three components. The expressions for evaluating each of these components have been derived. A similar analysis is then carried out for a protocol which is a variant of TDMA for buffered users and the superiority of the protocol employing

collision resolution algorithm has been demonstrated

Finally, the thesis has dealt with random access channels with multipacket reception capability. In such systems, even if two or more packets are transmitted in a slot, some of them may be received successfully. Code division multiple access–random access (CDMA–RA) is an example. In such systems, the users encode their packets with pseudo–orthogonal codes. When the coded packets are transmitted, then some may be received successfully. The unsuccessful packets may require retransmission. There are various schemes for assigning these codes. As pointed out in the thesis, transmitter based code assignment scheme with a multibit feedback broadcast by the common receiver is a generalization of the common receiver model of the finite user random access system.

We have studied this generalized common receiver model where each user is provided with a set of  $W$  orthogonal codes. A user, before transmitting its packet, encodes the packet with one of the codes to be specified by the collision resolution algorithm. We have considered blocked random access environment. It is also assumed that the users can generate only one packet per epoch. Then under the assumption of the number of active users at the beginning of a collision resolution epoch to be known, existence of an optimal algorithm is shown. The optimal control actions can be computed by the dynamic programming equations analogous to that of the conventional finite user random access model. These control actions specify the number of users to be enabled for each code. The algorithm has been illustrated with numerical examples for representative number of users.

From our studies, several interesting possibilities for future research emerge. Some of them can be outlined as follows:

- It would be of interest to investigate the relationship between the various classes of random access algorithms for the finite and the infinite user model.
- Our study is limited to the blocked random access algorithms. We have not considered other channel access schemes like free access. The determination of an optimal algorithm with free access is an interesting

area for future research

- We have shown in this thesis that when the number of active users is known only probabilistically, the optimal collision resolution problem can be formulated in principle as the first passage problem of a Partially Observable Markovian Decision Process. However, implementing this algorithm is computationally burdensome. The determination of optimal control actions with computationally feasible means is a topic for future research.
- We have considered two suboptimal algorithms SOA1 and SOA2. We have not investigated whether other suboptimal algorithms with better performance exist.
- In the case of buffered users, we have adopted a strategy for accessing the channel and analyzed this strategy. The determination of an optimal strategy can be pursued as a future course of investigation.
- We have given a rather brief look at the channels with multipacket reception capability. The thesis has considered only a limited case of transmitter based code assignment, since such a model can be shown to be a generalization of the common receiver model for the finite user random access system. It would be interesting to consider other code assignment schemes and compare them with the transmitter based code assignment in relation to its advantages and disadvantages. In this connection, we may use Time– Frequency code assignment scheme suggested in [13] for random access systems. A preliminary investigation in this direction has been carried out but not reported here.

# Appendix A

## Proof of Existence and Determination of Optimal Algorithm

---

In this appendix, following [11], we sketch the proof of Theorem 3.2 below

### Proof of Theorem 3.2

We have assumed the initial state to be  $S(0) = (N, n)$ . Consider the discounted cost

$$\begin{aligned}\Omega_{\Phi}(S(0) = (N, n), \beta) &= \sum_{t=0}^{\infty} \beta^t E_{\Phi}[\Psi(S(t), u(t))] \\ &= E_{\Phi} \sum_{t=0}^{\infty} \beta^t \Psi(S(t), u(t))\end{aligned}\tag{A.1}$$

As proved in [11], there exists a policy  $\tilde{\Phi} \in \mathcal{Y}_D$  such that  $\Omega_{\tilde{\Phi}}(S(0) = (N, n), \beta) \leq \Omega_{\Phi}(S(0) = (N, n), \beta)$  for all values of  $S(0) \in \mathcal{S}$  and  $\beta$  near one. Since  $\lim_{\beta \rightarrow 1} \Omega_{\tilde{\Phi}}(S(0) = (N, n), \beta) = \sum_{t=0}^{\infty} E_{\tilde{\Phi}}[\Psi(S(t), u(t))]$ , we have

$$\begin{aligned}C_{\tilde{\Phi}}(S(0) = (N, n)) &= E_{\tilde{\Phi}} \sum_{t=0}^{\infty} \Psi(S(t), u(t)) \\ &= \sum_{t=0}^{\infty} E_{\tilde{\Phi}}[\Psi(S(t), u(t))] \\ &\leq \sum_{t=0}^{\infty} E_{\Phi}[\Psi(S(t), u(t))] \\ &= E_{\Phi} \sum_{t=0}^{\infty} \Psi(S(t), u(t))\end{aligned}$$

$$= C_{\Phi}(S(0) = (N, n)) \quad (\text{A } 2)$$

This proves the theorem  $\square$

We now prove the Theorem 3.3 below. To prove this theorem, we first state the following lemma

**Lemma A.1** *If  $p_{(0,0),(0,0)}(u) = 1 \quad \forall u \in \mathcal{U}((0,0))$ ,  $S_i = (N_i, n_i)$  and  $p_{(N_i, n_i)(0,0)}^{(k)}(\Phi) > 0$  for some  $k \geq 1$ ,  $\Phi \in \mathcal{Y}_D$  and  $(N_i, n_i) \in \mathcal{S}$  then*

$$\lim_{t \rightarrow \infty} \sup_{\Phi \in \mathcal{Y}} \text{Prob}[S(t) \neq (0,0) \mid S(0) = (N_i, n_i)] = 0$$

### Proof of Theorem 3.3

This theorem states that let  $V_0(S_i) \forall S_i \in [\{S\} - (0,0)]$  be chosen arbitrarily. Define the following

$$V_{m+1}(S_i) = \min_{u \in \mathcal{U}(S_i)} \left[ \Psi(S_i, u) + \sum_{\substack{S_j \in \mathcal{S} \\ S_j \neq (0,0)}} p_{S_i, S_j}(u) V_m(S_j) \right] \quad (\text{A } 3)$$

then  $\lim_{m \rightarrow \infty} V_m(S_i) = C_{\Phi^*}(S(0) = (N_i, n_i)) = L^c(N_i, n_i)$  where  $\Phi^* \in \mathcal{Y}_D$  minimizes  $C_{\Phi}(S(0) = (N, n))$

Let  $V'_0(S_i) = C_{\Phi^*}(S_i = (N_i, n_i))$  for  $S_i \in [\{S\} - (0,0)]$ . Since  $\Phi^*$  is optimal and  $\Phi^* \in \mathcal{Y}_D$ , we have  $V'_m(S_i) = V'_0(S_i)$  for  $m = 1, 2, \dots$ . Let  $u_i \in \mathcal{U}(S_i)$  for  $S_i \in [\{S\} - (0,0)]$  minimizes the right hand side of Equation A.3, then we have for  $m = 0, 1, \dots$ ,

$$V'_{m+1}(S_i) - V_{m+1}(S_i) \leq \sum_{S_j \neq (0,0)} p_{S_i, S_j}(u_i) |V_m(S_j) - V'_m(S_j)| \quad (\text{A } 4)$$

Now assume that  $V_m(S_i)$  is the  $m$ th iterate of  $V'_0(S_i)$ . If  $u'_i \in \mathcal{U}(S_i)$  is the control action minimizing the right hand side of the equation

$$V_{m+1}(S_i) = \min_{u \in \mathcal{U}(S_i)} \left[ \Psi(S_i, u) + \sum_{\substack{S_j \in \mathcal{S} \\ S_j \neq (0,0)}} p_{S_i, S_j}(u) V_n(S_j) \right] \quad (\text{A } 5)$$

then we have for  $m = 0, 1, \dots$ ,

$$V_{m+1}(S_i) - V'_{m+1}(S_i) \leq \sum_{S_j \neq (0,0)} p_{S_i, S_j}(u'_i) |V_m(S_j) - V'_m(S_j)| \quad (\text{A } 6)$$

We thus have from Equation A 4 and A 6

$$|V_{m+1}(S_i) - V'_{m+1}(S_i)| \leq \max \left\{ \sum_{S_j \neq (0,0)} p_{S_i, S_j}(u_i) |V_m(S_j) - V'_m(S_j)|, \sum_{S_j \neq (0,0)} p_{S_i, S_j}(u'_i) |V_m(S_j) - V'_m(S_j)| \right\} \quad (\text{A } 7)$$

Let  $\Phi^m \in \mathcal{Y}_D$  be the policy which chooses the control  $u_i$  or  $u'_i$  for state  $S_i$  giving the maximum of right hand side of this equation. Hence

$$|V_{m+1}(S_i) - V'_{m+1}(S_i)| \leq \sum_{S_j \neq (0,0)} p_{S_i, S_j}(\Phi^m) |V_m(S_j) - V'_m(S_j)| \quad (\text{A } 8)$$

By repeatedly iterating this equation, we have

$$\begin{aligned} |V_{m+1}(S_i) - V'_{m+1}(S_i)| &\leq \sum_{S_j \neq (0,0)} \text{Prob}[S(m) = S_j \mid S(0) = S_i, \Phi^m] |V_0(S_j) - V'_0(S_j)| \\ &\leq \text{Prob}[S(m) \neq (0,0) \mid S(0) = S_i, \Phi^m] \max_{S_j} |V_0(S_j) - V'_0(S_j)| \end{aligned} \quad (\text{A } 9)$$

where  $\tilde{\Phi}^m \in \mathcal{Y}$  takes actions according to policy  $\Phi^{m-t}$  for  $0 \leq t \leq m$

By taking limits on both sides of this equation and applying Lemma A 1, we have

$$\lim_{m \rightarrow \infty} |V_{m+1}(S_i) - V'_{m+1}(S_i)| = 0 \quad (\text{A } 10)$$

Since  $V'_{m+1}(S_i) = V'_0(S_i) = C_{\Phi^*}(S(0) = S_i)$  Hence

$$\lim_{m \rightarrow \infty} V_{m+1}(S_i) = C_{\Phi^*}(S(0) = S_i)$$

This proves the theorem □

## Appendix B

### Probability of the Tagged Packet Delay

---

Let us choose a packet (that arrived in the arriving epoch of length  $j$ ) from the buffer of an active user, say  $a_k$ , and tag it. Let  $b_d^{a_k}$  denote the number of packets in the buffer of the active user  $a_k$  at the beginning of the departing epoch. Since  $a_k$  is an active user, we have  $b_d^{a_k} \geq 1$ . If  $\sigma$  is the arrival probability, then we have

$$Prob[b_d^{a_k} = m] = \frac{\binom{j}{m} \sigma^m (1 - \sigma)^{j-m}}{1 - (1 - \sigma)^j} \quad (\text{B.1})$$

Let  $\Pr[k]$  denote the probability the tagged packet (TP) has a delay of  $k$ . Then we have

$$\Pr[k] = \sum_{m=1}^j \Pr[k \mid b_d^{a_k} = m] Prob[b_d^{a_k} = m] \quad (\text{B.2})$$

where  $\Pr[k \mid b_d^{a_k} = m]$  is the probability that the TP has a delay of  $k$  given that  $b_d^{a_k} = m$ . Now,  $\Pr[k \mid b_d^{a_k} = m]$  can be written as

$$\begin{aligned} \Pr[k \mid b_d^{a_k} = m] = \\ \sum_{r=1}^m \Pr[k \mid b_d^{a_k} = m, \text{TP is } r\text{th packet}] Prob[\text{TP is } r\text{th packet in buffer} \mid b_d^{a_k} = m] \end{aligned} \quad (\text{B.3})$$

But  $Prob[\text{TP is } r\text{th packet in buffer} \mid b_d^{a_k} = m] = 1/m$ , hence we have

$$\Pr[k \mid b_d^{a_k} = m] = \frac{1}{m} \sum_{r=1}^m \Pr[k \mid b_d^{a_k} = m, \text{TP is } r\text{th packet}] \quad (\text{B.4})$$

We now illustrate how to evaluate  $\sum_{r=1}^m \Pr[k \mid b_d^{a_k} = m, \text{TP is } r\text{th packet}]$ . For this let us assume  $k = j - 1$ , i.e., the tagged packet has a delay of  $j - 1$ . By considering all possibilities of arrival pattern in the arriving epoch, it can be seen that

$$\begin{aligned} \sum_{r=1}^m \Pr[k = j - 1 \mid b_d^{a_k} = m, \text{TP is } r\text{th packet}] &= \frac{\binom{1}{0} \binom{j-2}{m-1} + \binom{1}{1} \binom{j-2}{m-2}}{\binom{j}{m}} \\ &= \frac{\binom{j-1}{m-1}}{\binom{j}{m}} \end{aligned} \quad (\text{B } 5)$$

By substituting this value in Equation B 4, we get

$$\begin{aligned} \Pr[k = j - 1 \mid b_d^{a_k} = m] &= \frac{1}{m} \frac{\binom{j-1}{m-1}}{\binom{j}{n}} \\ &= \frac{1}{j} \end{aligned} \quad (\text{B } 6)$$

Thus we have from Equation B 2

$$\Pr[k = j - 1] = \frac{1}{j} \quad (\text{B } 7)$$

For other values of  $k$  also, it can be easily verified that

$$\sum_{r=1}^m \Pr[k = j - 1 \mid b_d^{a_k} = m, \text{TP is } r\text{th packet}] = \frac{\binom{j-1}{m-1}}{\binom{j}{m}} \quad (\text{B } 8)$$

Thus we have  $\Pr[k] = \frac{1}{j}$  for all values of  $k$

□

# References

- [1] N Abramson The ALOHA system - another alternative for computer communication In *AFIPS conference proceedings*, pages 281–285, 1970
- [2] T Berger The Poisson multiple access conflict resolution problem In G Longo, editor, *Multi-User Communication System*, pages 1–26 Springer Verlag, 1980
- [3] T Berger and N Mehravari Conflict resolution protocols for secure multiple access communication system In G Longo, editor, *Secure Digital Communication*, pages 168–230 Springer Verlag, 1983
- [4] T Berger, N Mehravari, D Towsley, and J K Wolf Random multiple access communication and group testing *IEEE Transactions on Communication*, COM-32(7) 769–779, 1984
- [5] D P Bertsekas *Dynamic Programming Deterministic and Stochastic Models* Prentice Hall, 1987
- [6] J I Capetanakis Generalized TDMA The multi-accessing tree protocol *IEEE Transactions on Communication*, COM-37(10) 1476–1484, October 1979
- [7] J I Capetanakis Tree algorithms for packet broadcasting channels *IEEE Transactions on Information Theory*, IT-25 505–515, September 1979
- [8] I Cidon and M Sidi The effect of capture on collision resolution algorithm Technical Report EE-454, Technion– Israel Institute of Technology, Haifa, 1983
- [9] T M Cover and J A Thomas *Elements of Information Theory* John Wiley and Sons Inc , 1991
- [10] R Cruz and B Hajek A new upper bound on the throughput of a multi-access broadcast channel *IEEE Transactions on Information Theory*, IT-28(3) 402–405, 1982

- [11] C Derman *Finite State Markovian Decision Process* Academic Press, 1970
- [12] J N Diagle Throughput in asynchronous CDMA systems In *Proceedings of IEEE INFOCOM*, pages 1079–1089, 1987
- [13] G Einarsson Address assignment for a time–frequency coded spread spectrum system *The Bell Systems Technical Journal*, 59(7) 1241–1255, September 1980
- [14] G Fayolle, P Flajolet, M Hofri, and P Jacquet Analysis of a stack algorithm for random multiple access communication *IEEE Transactions on Information Theory*, IT-31(2) 244–254, 1985
- [15] D Freedman *Approximating Countable Markov Chains* Academic Press, 1972
- [16] R G Gallager Conflict resolution in random access broadcast networks In *Proceedings of the AFOSR Workshop in Communication Theory and Application*, pages 74–76, September 1978.
- [17] R G Gallager A perspective on multiaccess channels *IEEE Transactions on Information Theory*, IT-31(2) 124–142, March 1985
- [18] R G Gallager and D Bertsekas *Data Networks* Prentice Hall, second edition, 1992
- [19] E Gelenbe Performance analysis of multiuser communication system Access methods and protocols In G Longo, editor, *Multi–User Communication System* Springer Verlag, 1980
- [20] L Georgiadis, L F Merakos, and P Papantoni-Kazakos A method for the delay analysis of random multiple access algorithms whose delay process is regenerative *IEEE Journal on Selected Areas in Communications*, SAC-5(6) 1051–1062, July 1987
- [21] L Georgiadis and P Papantoni-Kazakos A collision resolution protocol for random access channels with energy detectors *IEEE Transactions on Communication*, COM-30(11) 2413–2420, November 1982
- [22] L Georgiadis and P Papantoni-Kazakos Limited feedback sensing algorithm for the packet broadcast channel *IEEE Transactions on Information Theory*, IT-31(2) 280–294, March 1985

- [23] L. Georgiadis and P. Papantoni-Kazakos. A 0.487 throughput limited sensing algorithm. *IEEE Transactions on Information Theory*, IT-33(2): 233–237, March 1987.
- [24] S. Ghez, S. Verdu, and S. Schwartz. Stability properties of slotted ALOHA with multipacket reception capability. *IEEE Transactions on Automatic Control*, AC-33(7): 640–649, July 1988.
- [25] S. Ghez, S. Verdu, and S. Schwartz. Optimal decentralized control in random access multipacket channel. *IEEE Transactions on Automatic Control*, AC-34(11): 1153–1163, November 1989.
- [26] D. J. Goodman, P. S. Henry, and V. K. Prabhu. Frequency hopped multilevel FSK for mobile radio. *The Bell Systems Technical Journal*, 59(7): 1257–1275, September 1980.
- [27] B. Hajek. Recursive retransmission control application to a frequency hopped spread spectrum system. In *Proceedings of Conference on Information Science and Systems CISS*, pages 116–120, 1982.
- [28] B. Hajek and Timothy van Loon. Decentralized dynamic control of a multiaccess broadcast channel. *IEEE Transactions on Automatic Control*, AC-27(3): 559–569, June 1982.
- [29] D. P. Heyman. Approximating the stationary distribution of an infinite stochastic matrix. In W. J. Stewart, editor, *Numerical Solutions of Markov Chains*. Marcel Dekker, 1991.
- [30] M. C. Hu and J. F. Chang. Collision resolution algorithm for CDMA systems. *IEEE Journal on Selected Areas in Communications*, SAC-8(4): 542–554, May 1990.
- [31] J. Huang and T. Berger. Delay analysis of interval searching contention resolution algorithm. *IEEE Transactions on Information Theory*, IT-32: 264–273, 1985.
- [32] P. A. Humblet. On the throughput of channel access algorithms with limited sensing. *IEEE Transactions on Communication*, COM-34(4): 345–347, April 1986.
- [33] Y. C. Jenq. On the stability of ALOHA systems. *IEEE Transactions on Communication*, COM-28: 1936–1939, 1980.

- [34] YC Jenq Optimal transmission control of slotted ALOHA system *IEEE Transactions on Communication*, COM-29 891–895, 1981
- [35] K Joseph and D Raychaudhari Stability analysis of asynchronous CDMA systems In *Proceedings of IEEE GLOBECOM*, pages 1740–1746, 1986
- [36] S S Kamal and S A Mahmoud A study of users' buffer variations in random access satellite channels *IEEE Transactions on Communication*, COM-27(6) 857–868, June 1979
- [37] L Kleinrock *Queueing Systems*, volume 1 Wiley Interscience, 1975
- [38] L Kleinrock and S S Lam Packet switching in multiaccess broadcasting channel *IEEE Transactions on Communication*, COM-23 410–423, 1975
- [39] S S Lam A new measure for characterizing data traffic *IEEE Transactions on Communication*, COM-26(1) 137–140, 1978
- [40] S S Lam Multiple access protocols In W Chou, editor, *Computer Communications*, volume 1, pages 114–155 Prentice Hall, 1983
- [41] S S Lam and L Kleinrock Packet switching in a multiaccess broadcast channels Dynamic control procedures *IEEE Transactions on Communication*, COM-23 891–904, September 1975
- [42] VC M Leung and R W Donaldson Effects of channel errors on the delay-throughput performance and capacity of ALOHA multiple access systems *IEEE Transactions on Communication*, COM-34(5) 497–501, May 1986
- [43] J Massey Channel models for random access systems In J K Skwirzynski, editor, *Performance Limits in Communication Theory and Practice*, pages 391–401 Kluwer Academic, 1988
- [44] J L Massey Collision resolution algorithms and random access communication In G Longo, editor, *Multi-User Communication System*, pages 73–137 Springer Verlag, 1980
- [45] P Mathys and P Flajolet Q-ary collision resolution algorithms in random access systems with free or blocked channel access *IEEE Transactions on Information Theory*, IT-32(2) 217–243, March 1985
- [46] R B Mattingly and C D Meyer Computing the stationary distribution vector of an irreducible Markov chain on shared memory multiprocessor In WJ Stewart, editor, *Numerical Solutions of Markov Chains* Marcel Dekker, 1991

- [47] N. Mehravari. Random access communication with multiple reception. *IEEE Transactions on Information Theory*, IT-36(3) 614–622, May 1990.
- [48] V.A. Mikhailov and B.S. Tsybakov. An upper bound on the capacity of random multiple access system. *Problemy Peredachi Informatsii*, 17(1) 90–95, January–March 1981.
- [49] K. Mittal and A.N. Venetsanopoulos. A note on input control policy for an ALOHA access scheme. *IEEE Transactions on Communication*, COM-39(2) 197–199, February 1991.
- [50] M.L. Molle. On the capacity of infinite population multiple access protocols. *IEEE Transactions on Information Theory*, IT-28(3) 396–401, 1982.
- [51] J. Mosley and P.A. Humblet. A class of efficient contention resolution algorithms for multiple access. *IEEE Transactions on Communication*, COM-33(2) 145–151, February 1985.
- [52] J.M. Musser and J.N. Diagle. Throughput analysis of an asynchronous code division multiple access system. In *Proceedings of IEEE ICC*, pages 2F2.1–2F2.7, 1982.
- [53] A. Nakasis and A. Ephremides. Steady state behavior of interacting queues: A numerical approach. In W.J. Stewart, editor, *Numerical Solutions of Markov Chains*. Marcel Dekker, 1991.
- [54] A.G. Pakes. Some conditions for ergodicity and recurrence of Markov chains. *Operations Research*, 17 1058–1061, 1969.
- [55] S.S. Panwar. On achieving a throughput of one for a random access channel with collision of known multiplicity. In *IEEE Symposium on Information Theory*, 1991.
- [56] N. Pippenger. Bounds on the performance of protocols for a multiple access broadcast channel. *IEEE Transactions on Information Theory*, IT-27 145–151, 1981.
- [57] L.K. Platzman. *Finite Memory Estimation and Control for Finite Probabilistic Systems*. PhD thesis, Massachusetts Institute of Technology, January 1977.
- [58] A. Polydoros and J. Silvester. Slotted random access spread spectrum networks: An analytical framework. *IEEE Journal on Selected Areas in Communications*, SAC-5(6) 989–1002, July 1987.

- [59] M B Pursley Frequency hop transmission for satellite packet switching and terrestrial packet radio networks *IEEE Transactions on Information Theory*, IT-33(5) 652–667, September 1986
- [60] D Raychaudhuri Performance analysis of random access packet switched code division multiple access system *IEEE Transactions on Communication*, COM-20(6) 895–901, June 1981
- [61] S Resheff and I Rubin Performance of a coded band-limited spread spectrum multiple access scheme using channel sensing *IEEE Journal on Selected Areas in Communications*, 8(4) 515–528, May 1990
- [62] R L Rivest Network control by Bayesian broadcast *IEEE Transactions on Communication*, COM-29(3) 891–895, May 1981
- [63] L G Roberts ALOHA systems with and without slots and capture. *Computer Communications Review*, 5 28–42, April 1975
- [64] S M Ross *Applied Probability Models with Optimization Applications* Holden Day, Inc , 1970
- [65] G Ruget Some tools for the study of channel sharing algorithms In G Longo, editor, *Multi-User Communication Systems* Springer Verlag, 1980
- [66] T Saadawi and A Ephremides Analysis, stability and optimization of slotted ALOHA with a finite number of buffered users *IEEE Transactions on Automatic Control*, AC-26(3) 680–689, June 1981
- [67] A Segall Recursive estimation from discrete point process *IEEE Transactions on Information Theory*, IT-22 422–431, 1976
- [68] M Sidi and Segall A Two interfering queues in packet radio networks *IEEE Transactions on Communication*, COM-31(1) 123–129, January 1983
- [69] M Sobel and PL Groll Group testing to eliminate efficiently all defectives in a binomial sample *The Bell Systems Technical Journal*, 38 1178–1259, September 1979
- [70] F D Sykas, D E Karvelas, and E N Protonotarios Queueing analysis of some buffered random multiple access schemes *IEEE Transactions on Communication*, COM-34(8) 790–798, August 1986
- [71] W Szpankowski Performance evaluation of a protocol for multiaccess systems In *Proceedings of Performance'83*, pages 337–395, 1983

- [72] J N Tsitsiklis Analysis of a multiaccess control scheme *IEEE Transactions on Automatic Control*, AC-32(11) 1017–1020, November 1987
- [73] B S Tsybakov Resolution of a conflict of known multiplicity *Problemy Peredachi Informatsii*, 16(2) 69–82, April–June 1980
- [74] B S Tsybakov Survey of USSR contributions to random multiple access communications *IEEE Transactions on Information Theory*, IT-31(2) 143–165, March 1985
- [75] B S Tsybakov Comparison of some multiple access systems *Problems of Information Transmission*, pages 57–62, 1990
- [76] B S Tsybakov and V B Faingol'd Blocked RMA stack algorithm for a network with a finite number of stations *Problemy Peredachi Informatsii*, 28 89–96, January–March 1992
- [77] B S Tsybakov and N B Likhanov Upper bound on the capacity of a random multiple access system *Problems of Information Transmission*, pages 224–236, 1988 Translated from *Problemy Peredachi Informatsii*, vol 23, no 3, pp 64–78, 1987
- [78] B S Tsybakov and V A Mikhailov Slotted multiaccess packet broadcasting feedback channel *Problemy Peredachi Informatsii*, 14 32–59, October–December 1978
- [79] B S Tsybakov, V A Mikhailov, and Likhanov N B Bounds to packet rate in multiple access channel *Problemy Peredachi Informatsii*, 19 61–81, January–March 1983
- [80] B S Tsybakov and V A Mikhailov Random multiple packet access Part and try algorithm *Problems of Information Transmission*, pages 305–317, 1981 Translated from *Problemy Peredachi Informatsii*, vol 16, no 4, pp 65–79, 1980
- [81] B S Tsybakov and N D Vvedenskaya Random multiple access stack algorithm *Problems of Information Transmission*, pages 230–243, 1981 Translated from *Problemy Peredachi Informatsii*, vol 16, no 3, pp 80–94, 1980
- [82] A J Viterbi Spread spectrum communications— myths and realities *IEEE Communications Magazine*, (5) 11–18, May 1979

- [83] N D Vvendenskaya and M S Pinsker Bounds on the capacity of FCFS multiple access algorithms *Problems of Information Transmission*, pages 49–56, 1990 Translated from Problemy Peredachi Informatsii, vol 26, no 1, pp 58–67, 1990
- [84] J E Wieseltheir and A Ephremides Some Markov chain problems in the evaluation of multiple access protocols In WJ Stewart, editor, *Numerical Solutions of Markov Chains* Marcel Dekker, 1991
- [85] J K Wolf Born again group testing Multiple access communications *IEEE Transactions on Information Theory*, IT-31(2) 185–191, March 1985
- [86] K C Wong and W Thanawastien Tree conflict resolution algorithms on multiple channel networks In *Proceedings of IEEE INFOCOM*, pages 276–285, 1987
- [87] M Yin and V O K Li Unslotted CDMA with fixed packet lengths *IEEE Journal on Selected Areas in Communications*, SAC-8(4) 529–541, May 1990

1801

1801

This book is to be returned on the  
date last stamped

IE-1994-D-KAR COL



A121801